Este exemplar corresponde à redação final da Tese/Dissertação devidamente corrigida e defendida por: ZANONI DIAS

e aprovada pela Banca Examinadora.

Campinas, 28 de MARÇO de 03

COORDENADOR DE PÓS-GRADUAÇÃO
CPG-IC

Rearranjo de Genomas:
Uma Coletânea de Artigos

Zanoni Dias

Tese de Doutorado

Instituto de Computação
Universidade Estadual de Campinas

Rearranjo de Genomas: Uma Coletânea de Artigos

Zanoni Dias[1]

14 de Novembro de 2002

Banca Examinadora:

- Prof. Dr. João Meidanis (Orientador)
- Profa. Dra. Maria Emília Machado Telles Walter
  Departamento de Ciência da Computação – CIC – UnB
- Prof. Dr. José Augusto Ramos Soares
  Departamento de Ciência da Computação – IME – USP
- Prof. Dr. Ricardo Dahab
  Instituto de Computação – Unicamp
- Prof. Dr. Pedro Jussieu de Rezende
  Instituto de Computação – Unicamp
- Prof. Dr. Carlos Eduardo Ferreira (Suplente)
  Departamento de Ciência da Computação – IME – USP
- Prof. Dra. Anamaria Gomide (Suplente)
  Instituto de Computação – Unicamp

[1]**Apoio financeiro da FAPESP, processo número 98/04432-0.**

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Dias, Zanoni

D543r Rearranjo de genomas: uma coletânea de artigos / Zanoni Dias – Campinas, [S.P. :s.n.], 2002.

Orientador : João Meidanis

Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Teoria da computação. 2. Algoritmos. 3. Evolução molecular. 4. Biologia molecular. I. Meidanis, João. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 14 de novembro de 2002, pela Banca Examinadora composta pelos Professores Doutores:

Profa. Dra. Maria Emília Machado Telles Walter
CIC - UNB

Prof. Dr. José Augusto Ramos Soares
IME - USP

Prof. Dr. Ricardo Dahab
IC - UNICAMP

Prof. Dr. Pedro Jussieu de Rezende
IC - UNICAMP

Prof. Dr. João Meidanis
IC - UNICAMP

Rearranjo de Genomas:
Uma Coletânea de Artigos

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Zanoni Dias e aprovada pela Banca Examinadora.

Campinas, 12 de Dezembro de 2002.

Prof. Dr. João Meidanis (Orientador)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Agradecimentos

Gostaria de agradecer a todas as pessoas que, direta ou indiretamente, me ajudaram nesta longa jornada, iniciada em 1994, quando ingressei no curso de Ciência da Computação da Unicamp, e que se encerra com esta tese. Sou grato a muitas pessoas, mas infelizmente, por falta de espaço, não poderei citá-las todas nominalmente aqui.

Primeiramente, gostaria de agradecer a todos os membros da banca. É uma grande honra poder contar com as contribuições de todos.

Fico particularmente feliz em poder defender esta tese diante de uma banca formada por pesquisadores como a Profa. Maria Emília Machado Telles Walter, com quem tive o prazer de trabalhar por vários anos. Esta parceria foi muito frutífera, resultando na publicação de cinco artigos, sendo que quatro deles integram esta tese. Mia, obrigado por tudo!

Agradeço à minha família, que sempre me deu apoio em todos os momentos. Mesmo distantes fisicamente, eles sempre estiveram do meu lado em todas as alegrias e tristezas. Quero agradecer especiamente aos meus pais, Antonia e Expedito, e às minhas irmãs, Marina e Lara.

Devo muito ao Prof. José Plinio de Oliveira Santos e à sua família, que me receberam de braços abertos nos meus primeiros tempos de Unicamp. Plinio foi meu primeiro orientador de iniciação científica. Durante aproximadamente dois anos trabalhei com teoria dos números no Departamento de Matemática Aplicada do IMECC-Unicamp.

Quero agradecer a Comissão de Pós-Graduação (CPG) por todo o apoio que recebi nestes quase cinco anos, em especial ao Prof. Pedro Jussieu de Rezende, que foi o principal responsável por me convencer a aceitar o desafio de ingressar direto no doutorado.

Acredito que uma pós-graduação é muito mais que uma tese. Neste sentido, considero que três fatores foram muito importantes para minha formação: os cursos de pós-graduação e as experências no LBI e no PED.

Tive a oportunidade de freqüentar mais de dez cursos de pós-graduação, todos muito interessantes, ministrados por alguns dos melhores professores que já tive o prazer de conhecer, como Arnaldo Vieira Moura, João Carlos Setubal, João Meidanis, Pedro Jussieu de Rezende, Ricardo Anido e Tomasz Kowaltowski.

Em 2000, participei, por dois semestres consecutivos, do Programa de Estágio Docente (PED). Esta foi uma experiência muito proveitosa que só foi possível graças aos professores

Flávio Keidi Miyazawa e João Meidanis, meus orientadores de PED, e Heloisa Vieira Rocha, que era a coordenadora da Comissão de Graduação do Instituto de Computação da Unicamp. Também quero agradecer aos quase cem alunos dos curso de Engenharia Elétrica e Engenharia Mecânica para os quais lecionei a matéria "Algoritmos e Programação de Computadores" (MC102).

Desde sua criação, fui um dos membros do Laboratório de Bioinformática (LBI) da Unicamp, o mais importante centro de apoio a projetos genomas do país. Através do LBI, participei de projetos importantes como o genoma da *Xylella fastidiosa* e o projeto EST de cana-de-açúcar (SUCEST). Agradeço aos coordenadores do LBI, João Carlos Setubal, João Meidanis e João Paulo Kitajima, por esta grande oportunidade, e aos colegas de laboratório, Vagner Katsumi Okura e Felipe Rodrigues da Silva, pelo companherismo em todos os dias e noites que trabalhamos juntos.

Durante anos, pertubei os funcionários e estagiários da secretaria de cursos do Instituto de Computação. Agora é a hora de agradecê-los: obrigado a todos pela paciência, em especial a Daniel de Jesus Capeleto e Vera Lúcia de Oliveira Ragazzi.

Em Campinas, tive a sorte de morar com bons companheiros, como Celmar Guimarães da Silva, Erasmo Gongorra Munuera, Thomaz Roberto Romano dos Reis e Yuri Lescow. Obrigado por me aturarem por todos estes anos.

Fiz grandes amigos na Unicamp, como Fabiano de Francisco Carlos, Grace Kelly de Castro Silva, Guilherme Pimentel Telles, José Augusto Amgarten Quitzau, Lin Tzy Li, Marília Dias Vieira Braga, Michel Cusnir, Naiade Viana Dourado, Patrícia Guimarães Takaki, Rui Kiyoshi Miadaira e Sônia Bannwart Santos. Pessoas muito especiais, que me ensinaram, no dia a dia, tanto quanto os melhores professores.

Duas pessoas, já mencionadas anteriormente, merecem alguns comentários especiais.

Quero agradecer a Lin Tzy Li, companheira de longa data, por tudo que ela tem feito por mim. Sem seu apoio, seu carinho e sua amizade sei que não teria chegado até aqui. Muito obrigado, meu amor. Te adoro!

Por último, gostaria de agradecer ao meu querido orientador João Meidanis. Desde o final do meu primeiro ano de graduação, quando ele me convidou pela primeira vez para trabalharmos juntos, tenho tentado justificar toda a confiança que ele deposita em mim. Apesar deste convite, só fui integrar o grupo de trabalho de Biologia Computacional quase dois anos depois, em meados de 1996. Após um ano e meio de iniciação científica, ingressei na pós-graduação, sempre sob sua orientação.

Mais do que um orientador, Meidanis é um grande professor (o melhor que já vi lecionando) e um grande amigo. Tem uma frase que expressa bem o que penso a respeito dele: "quando eu crescer, quero ser igual a ele". É isso, ele é como um pai pra mim. Meidanis, esta tese é para você, de coração.

Resumo

Hoje em dia, estão disponíveis, publicamente, uma imensa quantidade de informações genéticas. O desafio atual da Genômica é processar estes dados de forma a obter conclusões biológicas relevantes. Uma das maneiras de estruturar estas informações é através de comparação de genomas, que busca semelhanças e diferenças entre os genomas de dois ou mais organismos.

Neste contexto, a área de Rearranjo de Genomas vem recebendo bastante atenção ultimamente. Uma forma de comparar genomas é através da distância de rearranjo, determinada pelo número mínimo de eventos de rearranjo que podem explicar as diferenças entre dois genomas. Os principais estudos em distância de rearranjo envolvem eventos de reversões e transposições.

A presente coletânea é composta de oito artigos, contendo vários resultados importantes sobre Rearranjo de Genomas. Estes trabalhos foram apresentados em seis conferências, sendo uma nacional e cinco internacionais. Dois destes trabalhos serão publicados em importantes revistas internacionais e outro foi incluído como um capítulo de um livro.

Nossas principais contribuições podem ser divididas em dois grupos: um novo formalismo algébrico e uma série de resultados envolvendo o evento de transposição.

A nova teoria algébrica relaciona a teoria de Rearranjo de Genomas com a de grupos de permutações. Nossa intenção foi estabelecer um formalismo algébrico que simplificasse a obtenção de novos resultados, até hoje, muito baseados na construção de diagramas.

Estudamos o evento de transposição de várias formas. Além de apresentarmos resultados sobre a distância de transposição entre uma permutação e sua inversa, também estudamos o problema de rearranjo envolvendo transposições e reversões simultaneamente, construindo algoritmos de aproximação e estabelecendo uma conjectura sobre o diâmetro.

Usamos o formalismo algébrico para mostrar que é possível determinar a distância de fusão, fissão e transposição em tempo polinomial. Este é o primeiro resultado polinomial conhecido para um problema de rearranjo envolvendo o evento de transposição.

Por último, introduzimos dois novos problemas de rearranjo: o problema de distância sintênica envolvendo fusões e fissões, e o problema de transposição de prefixos. Para ambos apresentamos resultados significativos, que avançam o conhecimento na área.

Abstract

Nowadays, a huge amount of genetic information is publicly available. Genomic's current challenge is to process this information in order to obtain relevant biological conclusions. One possible way of structuring this information is through genome comparison, where we seek similarities and differences among the genomes of two or more organisms.

In this context, the area of Genome Rearrangements has received considerable attention lately. One way of comparing genomes is given by the rearrangement distance, which is determined by the minimum number of rearrangement events that explain the differences between two genomes. The main studies in rearrangement distance involve reversal and transposition events.

The present collection is composed of eight articles, containing several important results on Genome Rearrangements. These papers were presented in six conferences, one with Brazilian scope and five with international scope. Two of these works will be published in important international journals, and one other work appeared as a book chapter.

Our main contributions can be divided into two groups: a new algebraic formalism and a series of results involving the transposition event.

The new algebraic theory relates the genome rearrangement theory to the theory of permutation groups. Our intention was to establish an algebraic formalism that simplifies the creation of new results, up to now excessively based on the construction of diagrams.

We studied the transposition event in several ways. Besides presenting results on the transpositions distance between a permutation and its inverse, we also studied the rearrangement problem involving transpositions and reversals simultaneously, constructing approximation algorithms and proposing a conjecture on the diameter.

We used the algebraic formalism to show that it is possible to determine the distance of fusion, fission, and transposition in polynomial time. This is the first polynomial time result for a rearrangement problem involving the transposition event.

Finally, we introduced two now rearrangement problems: the syntenic distance problem involving fission and fusion, and the prefix transposition problem. For each one of these problems we present significant results, widening the knowledge in this area.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Com o advento de técnicas de seqüenciamento rápido, estamos testemunhando um crescimento espetacular na quantidade de dados moleculares, em especial, de DNA e seqüências protéicas. O grande desafio destes novos tempos da Genômica é processar esta enorme quantidade de dados e extrair informações biológicas relevantes.

Um modo de estruturar essas informações é através da comparação de genomas, através da qual nós analisamos dados de espécies distintas para descobrir similaridades e diferenças entre genomas relacionados. Entre os vários modos de se fazer este tipo de comparação, a área de Rearranjo de Genomas vem ganhando destaque ultimamente.

Nesta área, moléculas grandes de DNA, tipicamente extraídas de cromossomos, são analisadas considerando a ordem relativa de seus genes, com o objetivo de determinar a distância de rearranjo, determinada pelo número mínimo de eventos de rearranjo que podem explicar as diferenças entre as duas moléculas de DNA.

Muitos eventos de rearranjo foram recentemente estudados, tais como reversões, transposições, translocações, fusões e fissões, além de algumas combinações de eventos e alguns eventos restritos.

Uma reversão tem a propriedade de inverter a ordem de um conjunto de genes contíguos no genoma. Watterson e colegas [123] introduziram o primeiro algoritmo rudimentar para o problema. Kececioglu e Sankoff [78] apresentaram um algoritmo guloso que atingia uma aproximação de fator 2. Bafna e Pevzner [8] obtiveram um algoritmo de aproximação com fator $7/4$ para o problema usando uma estrutura denominada "Grafo de *Breakpoints*". Em 1998, Christie [29] mostrou um algoritmo de aproximação com fator $3/2$, sendo por muito tempo o melhor resultado conhecido. Berman, Hannenhalli e Karpinski [14] apresentaram recentemente um algoritmo de aproximação 1.375. Caprara [19] provou que o problema geral de ordenação por reversões é NP-Completo.

Uma série de outros trabalhos [22, 25, 15, 23, 24, 58, 74, 76, 77, 78] apresentaram abordagens alternativas para o problema levando em consideração aspectos mais práticos.

Uma das variações mais conhecidas deste problema é a ordenação por reversões quando as orientações dos genes são conhecidas. Bafna e Pevzner [8] descreveram um algoritmo de aproximação com fator $3/2$. Hannenhalli e Pevzer [64] provaram que esta variante do problema pode ser resolvida em tempo polinomial, apresentando um algoritmo de complexidade $O(n^4)$. Berman e Hannenhalli [13] posteriormente apresentaram um algoritmo com complexidade $O(n^2\alpha(n))$, onde $\alpha(n)$ é o inverso da função de Ackerman, que é uma função praticamente constante. Kaplan, Shamir e Tarjan [73] introduziram o melhor algoritmo conhecido para o problema, que roda em tempo $O(n^2)$. Meidanis, Walter e Dias [92] provaram que o problema de ordenação por reversões em cromossomos circulares pode ser resolvido, quando se conhecem as orientações dos genes, em tempo $O(n^2)$.

Na literatura de Rearranjo de Genomas encontramos ainda vários outros trabalhos [2, 4, 27, 20, 52, 66, 71, 117, 11, 12, 5] envolvendo variantes de problemas de distância de reversão.

Outro evento de rearranjo bastante estudado é a transposição, que troca dois conjuntos adjacentes de genes. Bafna e Pevzner [7, 9] descreveram um algoritmo de aproximação com fator $3/2$ para o problema, usando uma estrutura denominada "Diagrama de Ciclos". Christie [30] apresentou um algoritmo mais simples para o problema com o mesmo fator de aproximação, mas que tem uma complexidade de pior caso igual a $O(n^4)$. Alternativamente, Walter, Dias e Meidanis [121] apresentaram um algoritmo simples de ser implementado que roda em $O(n^2)$, mas cujo fator de aproximação é de 2.25. Guyer, Heath e Vergara [58] apresentaram uma série de heurísticas para este problema baseadas no comprimento da maior subseqüência e na maior subcadeia de uma permutação. Até o presente momento a complexidade do problema da distância de transposição permanece em aberto, não sendo conhecido nenhum algoritmo polinomial exato e nenhuma prova de que ele seja NP-completo.

Variantes do problema de distância de transposição, nas quais existem restrições quanto ao tamanho ou a posição dos blocos a serem trocados por uma transposição, foram estudadas por Aigner e West [3], Heath e Vergara [65] e por Dias e Meidanis [44].

Christie [28] descreveu um novo evento de rearranjo, denominado *block-interchange*, e que tem a propriedade de trocar dois conjuntos de genes, não necessariamente adjacentes. Desta forma, *block-interchange* pode ser considerado uma generalização do evento de transposição. Christie provou que o problema de se encontrar o menor número de *block-interchanges*, que transformam uma permutação em outra, pode ser resolvido em tempo quadrático.

Podemos mencionar ainda o evento de translocação, que possui a propriedade de trocar trechos de cromossomos distintos. As translocações foram alvo dos estudos de Hannenhalli [60] e Kececioglu e Ravi [75].

Outros dois eventos interessantes são a fusão e a fissão. Uma fusão atua sobre dois cromossomos unindo-os. Uma fissão, por sua vez, divide um cromossomo em dois. Dias e Meidanis [42] estudaram o problema de rearranjo de genomas envolvendo os eventos de fusão, fissão e transposição.

O problema da distância de rearranjo de genomas envolvendo simultaneamente reversões e transposições foi estudado por Blanchette, Kunisawa e Sankoff [16], Gu, Peng e Sudborough [55], Lin e Xue [85] e também por Walter, Dias e Meidanis [120, 96].

Chamamos de distância sintênica, o problema de se determinar a menor distância de rearranjo envolvendo genomas multi-cromossomais quando a ordem dos genes é desconhecida. Este problema foi estudado por Liben-Nowell [82, 83, 79, 84], DasGupta e colegas [38], Ferretti, Nadeau e Sankoff [48] e também por Dias e Meidanis [43].

Trabalhos relacionados a Rearranjo de Genomas foram tema de dissertações de mestrado, como as de Araujo [40], Curado [36] e Oliveira [41], como também de teses de doutorado, como as de Christie [30], Hannenhali [59], Vergara [117] e Walter [119].

Maiores informações sobre Biologia Computacional podem ser obtidos nos livros de Setubal e Meidanis [105, 106], Gusfield [57], Pevzner [103] e Waterman [122]. Informações adicionais sobre Rearranjo de Genomas podem ser obtidas nos artigos que compõem esta coletânea.

O restante deste capítulo será dedicado aos dois aspectos que compuseram este doutorado:

- O aspecto teórico, alvo principal desta tese, será tratado através da apresentação dos artigos que compõem esta coletânea (Seção 1.1);

- O aspecto prático será abordado brevemente na Seção 1.2, onde também descreveremos os trabalhos de bioinformática realizados pelo aluno no Laboratório de BioInformática (LBI) do Instituto de Computação da Unicamp.

1.1 A Coletânea de Artigos

Esta tese de doutorado é orientada a resultados em problemas de rearranjo de genomas, apresentados em forma de oito artigos, um artigo por capítulo. Teoricamente, o leitor pode ler qualquer um dos capítulos desta tese independentemente, já que todas as informações necessárias para a compreensão do trabalho estão definidas no próprio capítulo. O ponto negativo desta abordagem é a redundância de conceitos, sentida principalmente no início de cada capítulo.

Os resultados demonstrados nestes artigos foram apresentados em uma conferência nacional e em cinco conferências internacionais. Além disso dois trabalhos serão publicados em revistas internacionais, e outro foi incluído como um capítulo de um livro publicado em 2000.

Os artigos que compõem esta coletânea podem ser divididos em duas partes com quatro trabalhos cada. Na primeira encontram-se artigos cujos principais resultados foram apresentados anteriormente na tese de doutorado da Profa. Maria Emília M. T. Walter [119]. No segundo grupo, temos quatro trabalhos com contribuições originais. Nas duas seções seguintes descrevemos resumidamente cada um de nossos artigos.

1.1.1 Primeiros Trabalhos

Todos os trabalhos desta seção foram escritos em co-autoria com a Profa. Maria Emília M. T. Walter e com o Prof. João Meidanis. Estes trabalhos foram originalmente escritos entre 1997 e 1999, período que corresponde ao início desta tese e ao fim da tese de doutorado da Profa. Maria Emília M. T. Walter. Alguns desses trabalhos, como veremos adiante, foram revisados nos anos seguintes.

Nem todos os artigos escritos neste período fazem parte da presente coletânea. Este é o caso do trabalho "A New Approach for Approximating The Transposition Distance" (Maria Emília M. T. Walter, Zanoni Dias e João Meidanis) [121] apresentado no *String Processing and Information Retrieval (SPIRE'2000)* realizado na cidade de A Coruña, Espanha, no mês de setembro de 2000. Para maiores detalhes sobre os principais resultados deste trabalho ver Seção A.2.2.

Reversal Distance of Signed Circular Chromosomes

Neste nosso primeiro trabalho estudamos o problema de comparar dois cromossomos circulares, que evoluíram a partir de um ancestral comum através de reversões, supondo que são conhecidas a ordem dos genes e suas orientações. Apresentamos o primeiro algoritmo polinomial para determinar esta distância, baseado no algoritmo quadrático proposto por Kaplan, Shamir e Tarjan [72] para resolver o problema da distância de reversão de cromossomos lineares com sinais.

Além de calcular o diâmetro de reversão para cromossomos com sinais, tanto lineares quanto circulares, esclarecemos alguns pontos importantes sobre comparação de cromossomos lineares, inclusive corrigindo uma conjectura sobre o diâmetro de reversão para cromossomos lineares apresentada por Kececioglu e Sankoff [77] num trabalho de 1994.

Este artigo foi originalmente escrito em português e apresentado no *XXIV Brazilian Software and Hardware Seminars (SEMISH'97)*, realizado em Brasília, Distrito Federal, em agosto de 1997 [92]. No final do ano 2000, foi ampliado e inteiramente traduzido para o inglês. A versão aqui apresentada é a mesma que está depositada como relatório técnico no Instituto de Computação da Unicamp sob o número IC-00-23 [95].

Transposition Distance Between a Permutation and its Reverse

Neste trabalho, determinamos o menor número de transposições necessárias para transformar uma permutação qualquer na sua inversa. O problema tratado aqui foi proposto em 1995 por Bafna e Pevzner [7], no artigo que é considerado a principal referência sobre distância de transposição.

Provamos que a distância de transposição entre uma permutação de tamanho $n \geq 2$ e a permutação que representa a sua inversa éexatamente $\lfloor n/2 \rfloor + 1$. Apresentamos também um

algoritmo que calcula uma série ótima de eventos para este problema. Até hoje, este é um dos raros casos para o qual a distância de transposição é conhecida. Nós conjecturamos que este seja o valor do diâmetro de transposição.

Este artigo foi apresentado no *IV South American Workshop on String Processing (WSP'97)*, na cidade de Valparaíso, Chile, em novembro de 1997 [93].

Reversal and Transposition Distance of Linear Chromosomes

Neste artigo, apresentamos dois algoritmos de aproximação para o problema da distância de reversão e transposição para cromossomos lineares: um algoritmo com fator de aproximação 3 para o problema no qual as orientações dos genes são desconhecidas e outro com fator de aproximação 2 quando existe a informações sobre as orientações dos genes.

Nossos algoritmos baseiam-se no de fato que, no primeiro caso, sempre podemos remover pelo menos um *breakpoint* por operação, enquanto no segundo caso, garantimos que sempre é possível criar pelo menos um ciclo usando, uma reversão ou uma transposição.

Provamos também um limite inferior para o diâmetro de reversão e transposição para cromossomos quando conhecemos as orientações dos genes. Acreditamos que este valor seja de fato o diâmetro para estes eventos de rearranjo.

Em setembro de 1998 apresentamos este trabalho no congresso internacional *String Processing and Information Retrieval (SPIRE'98)*, em Santa Cruz de la Sierra, na Bolívia [120].

A Lower Bound on the Reversal and Transposition Diameter

Neste artigo, nós mostramos que a distância de reversão e transposição da permutação $\pi_n = (-1\ -2\ \ldots\ -(n-1)\ -n)$ em relação a identidade $\iota_n = (+1\ +2\ +3\ \ldots\ +(n-1)\ +n)$ é $\lfloor n/2 \rfloor + 2$ para $n \geq 3$, determinando assim um limite inferior não trivial para o diâmetro de reversão e transposições para genomas lineares com informações sobre a orientação de seus genes.

Este trabalho é uma extensão de alguns resultados apresentados no artigo anterior [120]. A versão apresentada aqui corresponde ao relatório técnico IC-00-16 de outubro de 2000 [94].

Alguns meses antes de terminarmos a redação deste relatório técnico submetemos um resumo para a revista *Journal of Computational Biology*. Após dois anos e meio, fomos comunicados pelos editores do *JCB* que este resumo foi aceito e será publicado no volume 9, número 5, de 2002 [96].

1.1.2 Contribuições Originais

Os quatro trabalhos resumidos a seguir foram desenvolvidos pelo proponente desta tese e seu orientador entre os anos de 2000 e 2002. Todos estes artigos são contribuições originais.

An Alternative Algebraic Formalism For Genome Rearrangements

Aqui relacionamos a recente teoria de Rearranjo de Genomas com a teoria de grupos de permutações de uma nova forma que acreditamos poder auxiliar em futuros avanços na área. Esse trabalho foi motivado pelo fato de muitos argumentos em Rearranjo de Genomas serem baseados em figuras, ou em enumeração exaustiva de todos os casos, evidenciando a falta de um formalismo algébrico adequado. Nós pretendemos dar à área de Rearranjo de Genomas um forte formalismo algébrico, assim como a geometria analítica forneceu alternativas a argumentos geométricos baseados em figuras.

Uma versão preliminar deste trabalho foi apresentada no congresso *Gene Order Dynamics, Comparative Maps and Multigene Families (DCAF'2000)*, realizado na cidade de Le Chantecler, no Canadá, em setembro de 2000 [89]. Este trabalho integra o livro *Comparative Genomics: Empirical and Analyitical Approaches to Gene Order Dynamics, Map Alignment and Evolution of Gene Families* [90] lançado em novembro do mesmo ano com os trabalhos apresentados no *DCAF'2000*.

Nesta tese, apresentamos uma versão estendida deste trabalho com resultados extras sobre o uso da teoria algébrica aplicada ao problema da distância de reversão.

The Genome Distance Problem by Fusion, Fission, and Transposition is Easy

Dados dois genomas circulares, fornecemos um algoritmo polinomial que determina uma série de fusões, fissões e transposições, de peso mínimo, que transforma um genoma no outro. Neste problema, fusões e fissões têm peso 1, enquanto que uma transposição recebe peso 2.

Este algoritmo é baseado em resultados clássicos da teoria de grupos de permutações e é o primeiro resultado polinomial para um problema de rearranjo de genomas envolvendo o evento de transposição.

A versão apresentada nesta tese corresponde ao relatório técnico número IC-01-07 do Instituto de Computação da Unicamp, de julho de 2001 [91]. Este trabalho, com pequenas alterações, foi apresentado no *String Processing and Information Retrieval (SPIRE'2001)* realizado na Laguna de San Rafael, no Chile, em novembro de 2001 [42].

The Genome Rearrangement Distance Problem with Arbitrary Weights

Este trabalho é uma continuação natural do artigo sobre distância de fusão, fissão e transposição apresentado no *SPIRE'2001* [42]. A diferença fundamental é que agora associamos peso 1 para fusões e fissões e um peso arbitrário para as transposições. Nós provamos que este problema é pelo menos tão difícil quanto o problema da distância de transposição, problema que ainda permanece em aberto.

Algumas variações do problema também são estudadas. Por exemplo, mostramos um algoritmo polinomial para o problema da distância sintênica com distinção de genes, quando apenas

eventos de fusões e fissões são permitidos. Provamos ainda que o problema similar de distância sintênica sem distinção de genes é NP-Difícil.

Estes resultados estão reunidos no relatório técnico IC-02-01 do Instituto de Computação de março de 2002 [43]. Pretendemos submeter nos próximos meses este trabalho a uma conferência internacional da área de Biologia Computacional.

Sorting by Prefix Transpositions

Neste trabalho, introduzimos um novo evento de Rearranjo de Genomas que denominamos de Transposição de Prefixos. Esta operação move o bloco formado pelos primeiros genes de um genoma linear para qualquer outra posição do genoma.

Apresentamos os primeiros resultados para o problema de ordenação por transposições de prefixos, como, por exemplo, algoritmos com fatores de aproximação 2 e 3. Conjecturamos que o diâmetro de transposição de prefixos é $D(n) = n - \left\lfloor \frac{n}{4} \right\rfloor$, e exibimos resultados de vários testes computacionais que sustentam esta hipótese.

Por último, propomos um algoritmo que decide quando uma permutação, que representa um genoma linear, pode ser ordenada usando apenas o número de transposições indicadas pela *lower bound* de *breakpoints*.

Estes resultados foram apresentados em Lisboa, Portugal, durante a realização do *String Processing and Information Retrieval (SPIRE'2002)*, em setembro de 2002 [44].

Este artigo foi selecionado para publicação pelos editores do *Journal of Discrete Algorithms*, presentes no SPIRE'2002. Uma versão estendida será especialmente preparada para a revista, cuja publicação deve ocorrer nos próximos meses.

1.2 Bioinformática

Um dos aspectos decisivos para o grande avanço da genética atual é o uso intensivo de técnicas computacionais. A *bioinformática* é a ciência responsável pela aplicação da informática para análise e administração de grandes quantidades de dados genéticos. De certa forma, podemos considerar a bioinformática como a faceta prática da biologia computacional.

Toda a pesquisa da moderna Biologia Molecular não seria possível sem a computação. Os programas de computador, por exemplo, têm papel fundamental na montagem de um genoma, a partir dos fragmentos de DNA obtidos pelos laboratórios de seqüenciamento. Além disso, são estes programas que permitem a comparação das seqüências descobertas com padrões genéticos já conhecidos e armazenados em bancos de dados.

Para se ter uma idéia do volume hoje (setembro de 2002) conhecido de informações genéticas, o GenBank [53], que é o maior banco de dados internacional de seqüências genéticas,

possui as seqüências completas de 1073 vírus, 99 procariotos e 17 eucariotos. E este número não pára de crescer, quase dobrando a cada ano.

E o Brasil, como um dos expoentes mundiais em genética, tem uma bioinformática de destaque. Boa parte deste reconhecimento se deve ao Laboratório de BioInformática (LBI) [80] do Instituto de Computação da Unicamp, responsável pela bioinformática dos primeiros projetos de seqüenciamento genético do país.

A seguir, descreveremos resumidamente dois importantes tipos de projetos em que o LBI esteve envolvido nos últimos anos: Projeto Genoma (Seção 1.2.1) e Projeto EST (Seção 1.2.2).

1.2.1 Projeto Genoma

Um projeto genoma tem como objetivo principal determinar as seqüências de DNA completas de todos os cromossomos de um organismo. A maior dificuldade é que as máquinas de seqüenciamento, hoje em dia, são capazes de obter, no máximo, mil bases de cada vez. Isto demonstra a inviabilidade de seqüenciar um cromossomo de um organismo complexo de uma só vez, já que, por exemplo, a seqüência de DNA de um cromossomo de uma bactéria é tipicamente formada por milhões de bases e a de um eucarioto pode chegar a cem milhões de bases.

Um dos métodos mais utilizados para se resolver este problema é o seqüenciamento de uma grande quantidade de fragmentos aleatórios de DNA, suficientes para cobrir o cromossomo original várias vezes. Assim, o problema se reduz a um grande quebra-cabeça que é resolvido por um programa de computador chamado montador, que reconstitui a molécula original, utilizando-se da sobreposição dos fragmentos. Montadores de genomas, devido a sua grande importância prática, são muito estudados [81, 26, 54, 98, 107].

Nos últimos anos, vários projetos genoma importantes foram concluídos, como é o caso da *Caenorhabditis elegans* [46] (verme), *Arabidopsis thaliana* [68] (planta), da *Drosophila melanogaster* [1] (mosca) e do *Homo sapiens* [32, 116] (homem). Em 1997, o Brasil, com o projeto genoma da *Xylella fastidiosa*, entrou para o seleto grupo de países com projetos genoma em andamento.

A *Xylella fastidiosa*, bactéria causadora da praga do amarelinho, foi o organismo escolhido para o primeiro projeto de seqüenciamento brasileiro principalmente por causa de três fatores: não haver no mundo, na época, outra pesquisa de seqüenciamento de uma bactéria que causa doenças em plantas (fitopatógeno); por sua importância econômica (cerca de 30% dos laranjais paulistas são afetados pelo amarelinho); e por causa do tamanho de seu genoma (2,7 milhões de pares de bases, relativamente pequeno).

A FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo [47], investiu cerca de US$ 15 milhões no genoma da *Xylella*, e boa parte deste valor foi utilizado para criar e equipar a Rede ONSA [100] (Organization for Nucleotide Sequencing and Analysis), formada por 35 laboratórios, sendo 34 de seqüenciamento e um de bioinformática (LBI), distribuídos pelo es-

tado de São Paulo. A Fundecitrus - Fundo de Defesa da Citricultura [51], e o CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico [31], também disponibilizaram recursos para este projeto.

O LBI foi o responsável por receber, via internet, os dados gerados pelos laboratórios de seqüenciamento e montar o genoma completo da *Xylella*, além de oferecer uma série de serviços para a comunidade de cientistas da Rede ONSA. Este trabalho de bioinformática foi realizado por uma equipe de onze pessoas e consumiu aproximadamente US$ 300 mil.

O artigo que descreve os resultados do projeto *Xylella* foi reportagem de capa da revista *Nature* [108], o que mostra a grande repercussão internacional alcançada pela iniciativa brasileira. O projeto determinou que o cromossomo principal da bactéria é formado por 2.679.305 bases, sendo identificados 2.838 genes. A *Xylella* possui ainda dois plasmídeos: o maior é formado por 51.158 bases e possui 65 genes, e o menor é formado por apenas 1.285 bases com 2 genes identificados. Maiores detalhes sobre a bioinformática do projeto *Xylella* podem ser obtidos na dissertação de mestrado de Okura [99].

Os 192 cientistas, entre eles o aluno e dez colegas do LBI, que fizeram da *Xylella* um grande sucesso, foram premiados em 21 de fevereiro de 2000, com a medalha do Mérito Científico e Tecnológico [88] instituída por Mário Covas, então governador do Estado de São Paulo.

O seqüenciamento do genoma da *Xylella* foi apenas a primeira etapa de um longo processo, cujo resultado final será a cura da praga do amarelinho. Pensando nisso, a FAPESP criou o Projeto Genoma Funcional [50], que desde o fim do seqüenciamento investiga os mecanismos de transmissão e de estabelecimento da bactéria na planta.

No rastro do sucesso da *Xylella*, o Brasil desenvolveu com sucesso outros projetos genoma como, por exemplo, o projeto das bactérias *Chromobacterium violaceum* realizado pelo Projeto Genoma Brasileiro [17] e o Projeto *Xanthomonas* [37] também realizado pela Rede ONSA.

1.2.2 Projeto EST

Projetos ESTs são alternativas aos projetos de seqüenciamento de genomas completos. Ao invés de seqüenciar todo o genoma de um organismo e depois tentar descobrir quais são seus genes, num projeto EST (*Expressed Sequence Tag*), apenas os genes expressos pelo organismo são capturados e seqüenciados.

Um EST, de forma geral, não corresponde a um gene inteiro e sim, a apenas uma parte dele. Surge, então, um importante problema computacional denominado *clustering*, ou seja, agrupar todos os ESTs que correspondem ao mesmo gene em um único grupo (*cluster*). Assim como no caso de montagem de genoma, vários métodos novos para *clustering* foram propostos recentemente, quase sempre baseados em experiências práticas em projetos de ESTs, como é o caso do trabalho de Miller e colegas [97] e Telles e da Silva [112].

O dbEST [39] (*Expressed Sequence Tags database*) é o maior banco público do mundo de

EST e conta, hoje em dia, com mais de 12 milhões de ESTs, entre eles, 4,5 milhões de ESTs humanos, 2,6 milhões de ESTs de camundongo, 350 mil ESTs de rato e 270 mil ESTs de soja. A partir de informações contidas em bancos como este, é possível deduzir a funcionalidade de um novo gene identificado num projeto EST. Há três anos, o dbEST possuia pouco mais de 2 milhões de ESTs depositados.

Mais ou menos na mesma época, em julho de 1999, a Rede ONSA ingressou em uma nova empreitada: o Projeto SUCEST (Sugarcane EST Project) [110], que tinha o ambicioso objetivo de ser o maior projeto público do mundo de seqüenciamento de ESTs de planta. A cana-de-açúcar foi escolhida por ser uma planta economicamente importante para o Brasil, responsável por 25% da produção mundial, que se concentra principalmente em países tropicais.

A cana-de-açúcar produzida hoje no Brasil é um híbrido de outras cinco espécies. Por causa disso, seu genoma é muito complexo, formado por um número variável de cromossomos (entre 70 e 120), o que inviabiliza seu estudo através de um projeto genoma convencional. Este projeto foi financiado pela FAPESP e pela Copersucar - Cooperativa de Produtores de Cana, Açúcar e Álcool do Estado de São Paulo [33]. O projeto SUCEST produziu 291.904 ESTs de cana-de-açúcar.

Nesse projeto, o Laboratório de BioInformática criou o *web site* que foi o "ponto de encontro" dos 74 laboratórios de seqüenciamento e análise que fizeram parte do consórcio.

O trabalho "Bioinformatics of the sugarcane EST project" (Guilherme P. Telles, Marília D. V. Braga, Zanoni Dias, Lin Tzy Li, José A. A. Quitzau, Felipe R. da Silva e João Meidanis) [111] descreve todas as atividades desenvolvidas pelo LBI, incluindo categorização, genômica comparativa e o *clustering* responsável por agrupar os ESTs em 43.141 *clusters*. Este trabalho foi publicado em dezembro de 2001, na edição especial da revista *Genetics and Molecular Biology* sobre o Projeto SUCEST. Nos próximos meses deve ser publicado um artigo com as principais contribuições biológicas deste projeto [118].

Logo após o início do Projeto SUCEST, surgiu o Projeto Câncer [113] desenvolvido pela Rede ONSA em parceria com o Instituto Ludwig de Pesquisa sobre o Câncer [86]. Esse projeto teve como objetivos gerar entre 500 mil e um milhão de ESTs oriundos de células humanas com câncer, produzindo aproximadamente 50 mil clusters. Outro projeto EST em desenvolvimento é o do parasita *Schistosoma mansoni* [104] que deve gerar cerca de 120 mil ESTs até o final de 2002.

Parte I

Primeiros Trabalhos

Capítulo 2

Reversal Distance of Signed Circular Chromosomes *

João Meidanis

Maria Emilia M. T. Walter

Zanoni Dias

Abstract

We study the problem of comparing two circular chromosomes, evolved from a common ancestor by reversals, given the order of the corresponding genes and their orientations. Determining the minimum number of reversals between the chromosomes is equivalent to look for the minimum number of reversals that transforms a circular sequence of signed integer numbers, defined in an appropriate manner, into another, where a reversal acts on a subsequence, reversing its order and flipping the signs. We carefully formalize the concepts of circular chromosome and circular reversal, and show that this problem is essentially equivalent to the analogous problem on linear chromosomes. As a consequence we derive polynomial time algorithms based on this observation. We also compute the reversal diameter for signed chromosomes, both linear and circular.

* *Trabalho depositado como Relatório Técnico no Instituto de Computação da Unicamp em dezembro de 2000, sob o número IC-00-23. Uma versão anterior deste relatório foi apresentado no* **XXIV Brazilian Software and Hardware Seminars** *(SEMISH'97), realizado em Brasília, Distrito Federal, em agosto de 1997.*

2.1 Introduction

The huge amount of data resulting from genome sequencing in Molecular Biology is giving rise to an increasing interest in the development of algorithms for comparing genomes of related species. Particularly these data allowed studies on mutational events acting on large portions of the chromosomes, that can be used to compare genomes for which the traditional methods of comparing DNA sequences are not conclusive. There are several mutational events affecting large fragments of genomes of organisms, and among them, the *reversal* seems to be one of the commonest. A reversal replaces a sequence of an arbitrary region of the chromosome with the reverse complementary sequence. This reverses the gene order within the region, and changes the orientation of each gene. In this paper we study the comparison of two genomes, formed each by a single circular chromosome, on the basis of the order and orientation of their common genes, and in terms of the mutational event of reversal.

A circular chromosome can be seen as a circular arrangement of blocks of genes, where each block has an *orientation*. Figure 2.1 shows examples of circular chromosomes of two species of plants, where each number represents a block composed by one or more genes, and the arrows indicate the orientations of the blocks of one species relative to the other.

In a circular chromosome, a reversal is defined by fixing two cut points in this chromosome, and reversing the order of the genes in one of the two regions delimited by these points (see Figure 2.2).

In general terms, the *problem of reversal distance of signed circular chromosomes* is formulated as follows. Given two circular chromosomes A and B, we want the shortest series of reversals that transforms A into B. This minimum number of reversals is called *reversal distance* between A and B. Figure 2.3 shows an example of a circular chromossome transformed into another with the minimum number of reversals.

Another version of this problem arises when the orientations of the genes on the chromosomes are not known. In that case, we have the *unsigned* version of the problem, where the reversals only reverse gene order. There are other versions of the same problem considering linear chromosomes, and other mutational events besides reversal. The literature on problems originated by different types of mutational events is growing very quickly in recent years. In the following, we briefly review other works studying reversal, observing that chromosomes are commonly represented by permutations in this context.

With respect to linear chromosomes, Aigner and West [3] had studied the problem of sorting a permutation, considering the operation of reinsertion of the first element in the sequence of the permutation. The sorting diameter (the maximum distance between two permutations) in this case is $n - 1$, where n is the number of elements of the permutation. Kececioglu and Sankoff [76, 78] had studied the problem of the reversal distance of unsigned linear permutations, and developed the first approximation algorithm for the problem. Their algorithm runs in

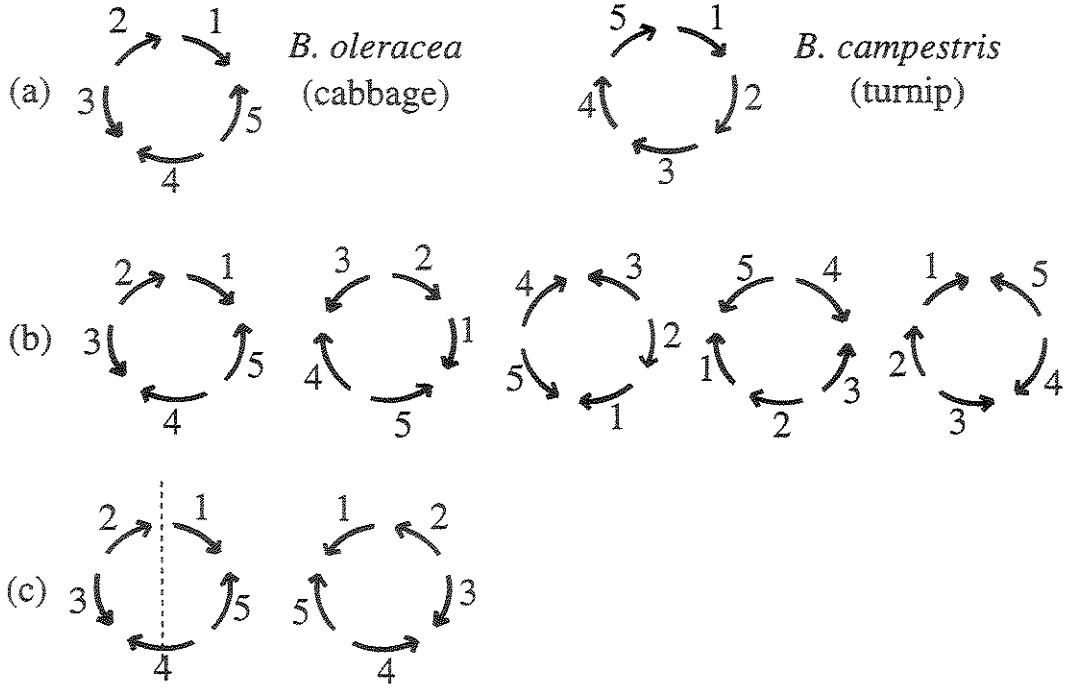

Figure 2.1: Examples of circular chromosomes of two species of plants. (a) The arrows indicate the orientations of a species relative to the other. (b) These examples show different representations of the same chromosome. (c) These examples show the same chromosome, considering the two possible forms to view the gene blocks of a circular chromosome. These two forms are considered equivalent, and these two chromosomes are obtained from one another by reflection relative to the axis shown in the figure.

$O(n^2)$ time and is guaranteed to use no more than two times the reversal distance. They also developed efficient bounds, used on a branch-and-bound algorithm, that solved to optimality or almost optimality permutations ranging from 30 to 50 elements. Bafna and Pevzner [8] afterwards introduced a new structure, the *breakpoint graph* of an initial permutation relative to a target permutation, that allowed to set up a more precise lower bound to the reversal distance, considering another parameter, based on a maximum alternating cycle decomposition, denoted by $c(\pi)$. Based on that graph, they devised an approximation algorithm with a performance guarantee of $7/4$, and introduced an approximation algorithm for signed permutations with a guarantee of $3/2$.

Hannenhalli and Pevzner [64] introduced two new parameters: the *number of hurdles* ($h(\pi)$) and an indicator of whether the breakpoint graph *is a fortress*, ($f(\pi)$). Together with the maximum number of cycles of the alternating cycles decomposition ($c(\pi)$) of the breakpoint graph, these parameters allowed the authors to demonstrate a *duality theorem.* Based on this theorem, they presented the first polinomial algorithm for the problem of the reversal distance of the

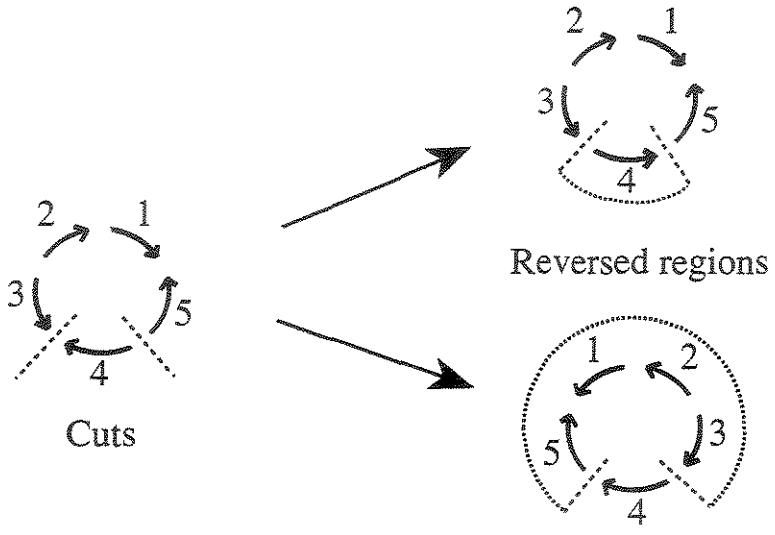

Figure 2.2: This example shows the two possibilities for reversal in a circular chromosome, given two cuts.

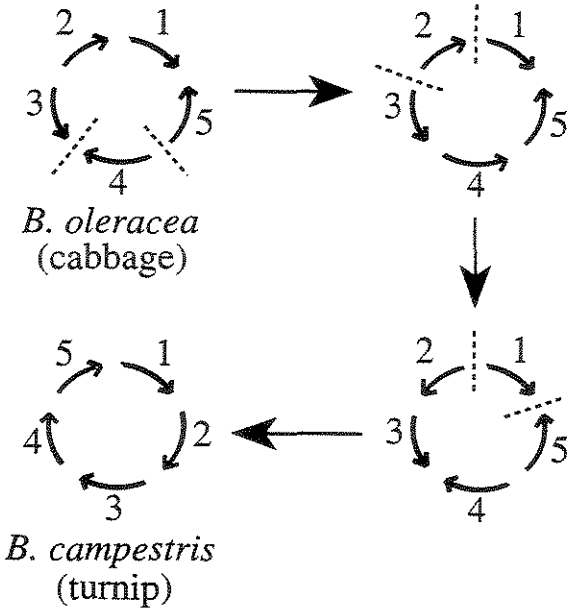

Figure 2.3: This example shows a series of reversals that transforms *B. oleracea* (cabbage) into *B. campestris* (turnip).

signed linear permutations, with time complexity $O(n^4)$. Berman and Hannenhalli [13] introduced new data structures on that algorithm and lowered the complexity to $O(n^2\alpha(n))$. Finally, Kaplan, Shamir and Tarjan [73], based on the Hannenhalli and Pevzner theory, and using part of the Berman and Hannenhalli algorithm, showed a new algorithm with $O(n^2)$ complexity. We will call this last one *KST algorithm.*

With respect to circular chromosomes, Watterson and other authors [123] showed an algorithm, very simple, to find out the reversal distance of circular permutations, establishing a lower bound $(number\ of\ breakpoints/2)$, and an upper bound $(n-2)$ for the reversal distance. They presented also a stochastic algorithm for the problem. Kececioglu and Sankoff [77] presented an exact branch-and-bound algorithm for the problem of reversal distance of signed circular permutations. This algorithm, using simple methods to find the lower and upper bounds, found extremely precise values for the reversal distance in several experiments. The authors reported that they did not know reasons to justify the proximity of these limits. Now we know that the Hannenhalli and Pevzner theory justifies these results, because $h(\pi)$ and $f(\pi)$ are small for random permutations.

In this paper we present a formalism for circular chromosomes and for reversals acting on them. As a consequence we show polinomial algorithms for the problem of reversal distance of signed circular chromosomes. These algorithms are based on the theory for the linear problem given by Hannenhalli and Pevzner [64]. Besides, we calculate the reversal diameter for linear and circular chromosomes.

In Section 2 we first formalize a circular chromosome by an equivalence class, and next we show that there is an isomorphism between reversals acting on circular chromosomes and reversals acting on linear chromosomes. This result allow us to compute the reversal distance of signed circular chromosomes by computing the reversal distance of signed linear chromosomes with one less gene. In Section 3 we show some results concerning the reversal distances of signed circular and linear permutations of the same size. In Section 4 we calculate the reversal diameter of signed linear and circular permutations. Finally, the last section brings conclusions of this work and indicates some future directions.

2.2 A formalization for the problem

2.2.1 Linear Chromosomes

We begin by presenting a brief overview of some important results about signed linear chromosomes, due mainly to Bafna and Pevzner [8] and Hannenhalli and Pevzner [64]. A signed linear chromosome is represented by a signed permutation. A *signed permutation* is an ordinary permutation, except that each element has positive $(+)$ or negative $(-)$ sign, indicating the relative orientation of the block. In this case, a *reversal* ϱ of the interval $[i, j]$ is denoted by $\varrho(i, j)$ and

we have

$$\varrho(i,j)\cdot\pi = (\pi_1\ldots\pi_{i-1}\overline{\pi}_j\overline{\pi}_{j-1}\ldots\overline{\pi}_{i+1}\overline{\pi}_i\pi_{j+1}\ldots\pi_n)$$

where $\overline{\pi}_k$ indicates the inversion of the sign of π_k.

The problem of the reversal distance of signed linear chromosomes is commonly formalized as follows. Given two permutations π and σ modeling two signed linear chromosomes, the *reversal distance problem* of π and σ is to find a series of reversals $\varrho_1, \varrho_2, \ldots, \varrho_t$ such that $\varrho_t \cdot \varrho_{t-1} \cdot \ldots \cdot \varrho_2 \cdot \varrho_1 \cdot \pi = \sigma$ and t is minimum. We call t the **reversal distance of π and σ**, denoted by $d(\pi,\sigma)$.

The algorithms of Bafna and Pevzner [8] and of Hannenhalli and Pevzner [64] are based on a structure called *breakpoint graph*. This graph is constructed from π and σ as follows. Each one of the signed integers is represented by an arrow, from left to right when the sign is $+$, and from right to left when the sign is $-$. The initial and final points of these arrows are the vertices of this graph. Besides, we add two reference points, one on the left of the sequence (labelled by L) and the other on its right (labelled by R). After that, we put *reality* edges joining extreme points of adjacent arrows in π, and *desire* edges joining extreme points of adjacent arrows in σ. Important properties of this graph are:

1. The resulting graph is formed by a collection of even cycles. When $\pi = \sigma$, the number of these cycles gets its maximum value, $n+1$. For two different permutations, there are less than $n+1$ cycles.

2. Each reality edge from a cycle whose size is larger than 2 represents a *breakpoint* in the permutation, that is, a point where a reversal will have to act in order to transform π into σ. When two vertices belong to a cycle of size 2, that is, are joined by two parallel edges, exactly one reality and one desire edge, we say that *there is not* a break in that position.

From this graph we can compute three parameters that allow us to compute the reversal distance of π and σ: the number of cycles $c(\pi,\sigma)$, the number of hurdles $h(\pi,\sigma)$ and a parameter $f(\pi,\sigma)$ indicating whether the graph is a fortress, where this last value can be equal to 1 or 0 only. The reversal distance is then given by:

$$d(\pi,\sigma) = n + 1 - c(\pi,\sigma) + h(\pi,\sigma) + f(\pi,\sigma).$$

We refer the readers to the important works [64, 13, 73] or to the introdutory text of Setubal and Meidanis [106] for a more detailed explanation on these parameters. We will not need details on hurdles and fortresses until Section 2.4, where we will review these concepts.

2.2.2 Circular Chromosomes

Now we give a formalization of a circular chromosome by an equivalence class.

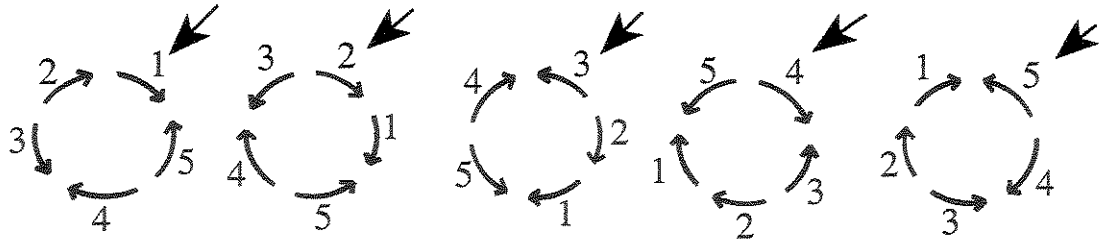

Figure 2.4: In a circular chromosome we can choose each one of the genes block as the first one. Then, all of these sequences are considered equivalent, and they represent the circular chromosome of *B. oleracea* shown in the Figure 1 (a).

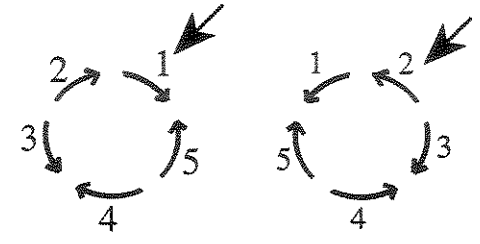

Figure 2.5: In a circular chromosome two sequences where one of them is obtained from the other by reflection are considered equivalent. The circular chromosome represented is *B. oleracea* shown in Figure 1 (a).

Intuitively, a circular chromosome is a circular arrangement of signed blocks (see Figure 2.1). A *block* of genes of the chromosome will be modelled by a signed integer. The sign "+" indicates an arrow in clockwise direction in Figure 2.1, and the sign "−" indicates an arrow in counterclockwise direction. Given an *initial block*, we can represent a circular chromosome by a sequence as follows. Through convention, we always read the blocks in clockwise direction. Walk around the chromosome in clockwise direction, beginning at the initial block, and write down the signed integers corresponding to the blocks found. Then, $\pi = (\pi_1 \pi_2 \dots \pi_n)$ will denote the circular chromosome, with n blocks of genes. As an example, the chromosome of *B. oleracea* of Figure 2.1a can be represented by the sequence $(+1\ -5\ +4\ -3\ +2)$.

We can choose each one of the blocks as the first one, and therefore we can have many different sequences representing the same chromosome (see Figure 2.4). All of these sequences are considered equivalent. Besides, two sequences where one of them is obtained by the other by reflection are considered equivalent, and in particular $\pi = (\pi_1 \pi_2 \dots \pi_n)$ and $s \cdot \pi = (\overline{\pi}_n \overline{\pi}_{n-1} \dots \overline{\pi}_2 \overline{\pi}_1)$ are considered equivalent sequences (see Figure 2.5).

This way, a sequence modeling a circular chromosome is a representative of an equivalence

class in the set of all sequences. Below we define the *rotation* and the *reflection* operations, that will formalize the two characteristics described above. From these operations we will define an equivalence relation between two sequences, and an equivalence class that will represent a circular chromosome.

We will call S_n the set of all possible sequences of distinct signed integers, where each sequence has size n. These integers must belong to the interval $[1..n]$. Observe that $|S_n| = 2^n n!$. Let us take $\pi = (\pi_1 \pi_2 \dots \pi_n)$, a sequence of S_n. We will define two types of operations acting in π as follows:

- *Rotations.* We will denote by r the basic rotation that moves the permutation elements one position to the left:

  $$r \cdot \pi = (\pi_2 \pi_3 \dots \pi_n \pi_1).$$

  We will define r^i for every $i \in Z$ in the usual way: r^i is the composition of r i times for $i > 0$ and r^{-i} is the inverse of r^i. Besides, r^0 is the identity. We have the following important relations:

  $r^n = r^0$, or more generally, $r^i = r^j$ if $i \equiv j \pmod{n}$ for all $i, j \in Z$.

  $r^i r^j = r^{i+j}$ for all $i, j \in Z$.

  The operations r^i are called *rotations*.

- *Reflections.* We will denote by s the basic reflection that inverses the order of the permutation and also the signs. So,

  $$s \cdot \pi = (\overline{\pi}_n \overline{\pi}_{n-1} \dots \overline{\pi}_2 \overline{\pi}_1).$$

  We will define s^i for all $i \in Z$ as follows: s^i is the composition of s i times for $i > 0$ and s^{-i} is the inverse of s^i. Note that $s^{-i} = s^i$. Besides, s^0 is the identity. We have the following important relations:

  $s^2 = s^0$, or more generally, $s^i = s^j$ if $i \equiv j \pmod{2}$ for all $i, j \in Z$.

  $s^i s^j = s^{i+j}$ for all $i, j \in Z$.

  We can apply r and s to a sequence, using the above definitions. Then, $rs\pi = r(s\pi) = r(\overline{\pi}_n \overline{\pi}_{n-1} \dots \overline{\pi}_2 \overline{\pi}_1) = (\overline{\pi}_{n-1} \overline{\pi}_{n-2} \dots \overline{\pi}_2 \overline{\pi}_1 \overline{\pi}_n)$.

  We have the following relation:

  $$rs = sr^{-1}. \tag{2.1}$$

  Generically, the operations sr^i are called *reflections*. Each reflection is equal to its own inverse.

Now we will define an equivalence relation between two sequences π and γ.

Definition 2.2.1 *Given two sequences π and γ, we define*

$$\pi \sim \gamma$$

if and only if there are $i, j \in Z$ such that $\gamma = r^i s^j \cdot \pi$.

The above relation is an equivalence relation. The proof of this result is simple. Equation (2.1) can be used in this proof.

From this equivalence relation, we can define an equivalence class of the sequence π, denoted by $[\pi]$, which represents a signed circular chromosome, as follows

$$[\pi] = \{\gamma \in S_n | \pi \sim \gamma\}$$

This formalization is interesting biologically, because it does not fix the first element of the sequence, and then each one of the genes block can be the first, it is sufficient to apply rotation. Besides, two sequences where one of them is obtained from the other by reflection can be produced applying the s operator.

2.2.3 Circular Reversals

We model now how a reversal will act in a class A representing a circular chromosome. First we note that there are two possibilities for a reversal acting on a circular chromosome, given the two points where the cuts have occurred (see Figure 2.6).

Suppose the two cuts occur between $i \ominus 1, i$ and $j, j \oplus 1$, with $1 \leq i \leq j \leq n$. Here $\ominus$ and $\oplus$ are the usual operations of subtraction and addition, respectively, except that we take the result modulo n and choose n rather than zero as the representative of the class of multiples of n. We will assume that these cuts are distinct, therefore $i \neq (j \oplus 1)$.

Also if we choose i and j such that $i > j$, we can change i and j without problems because both are just pointers to the cuts.

Then we have the following lemma.

Lemma 2.2.1 *Given a sequence π from an equivalence class A which models a circular chromosome, and two integers i and j with $1 \leq i \leq j \leq n$ and $i \neq (j \oplus 1)$ such that these cuts occur between $i \ominus 1, i$, and $j, j \oplus 1$, the sequences resulting from the two possible ways of reversing the circular chromosome between these cuts belong to the same equivalence class.*

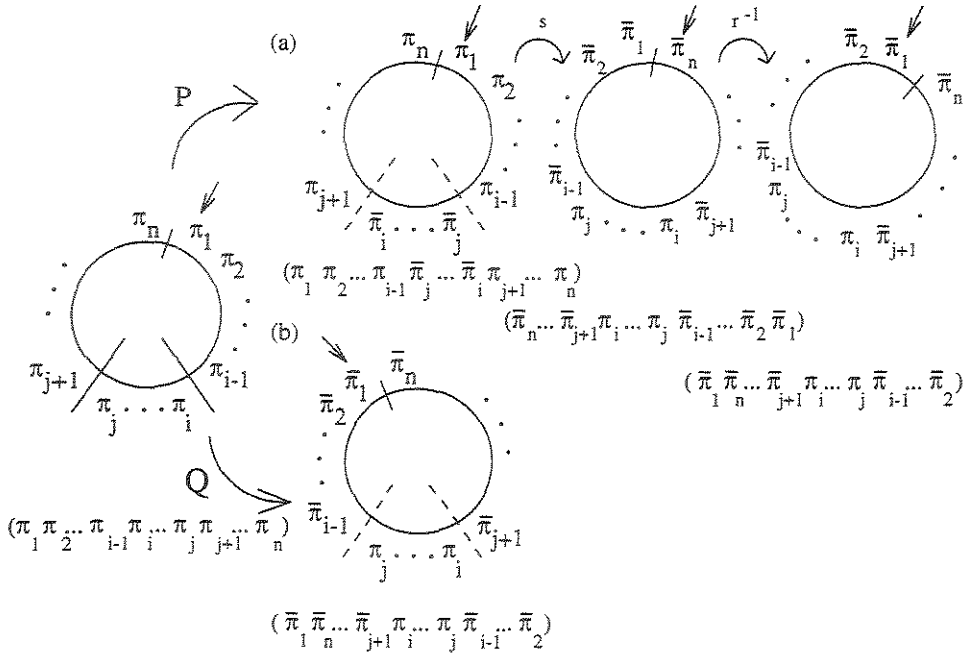

Figure 2.6: This figure shows that the two circular chromosomes resulting from the reversal are represented by two sequences that belong to the same equivalence class. Note that the arrow, before the reversion, indicates the first block of the sequence chosen from the equivalence class which represents the circular chromosome. The portion of the chromosome suffering the reversal can include or not the arrow. (a) In this case, the reversal does not include the arrow. The sequence resulting from the reversal is shown. (b) In that case, the reversal includes the arrow. We can apply reflection and rotation in the sequence resulting from the reversal in order to obtain the same sequence as in case (a). The sequences resulting from each operation are shown.

Proof:

We will denote by P and Q the two possible ways of reversing the circular chromosome (see Figure 2.6). Taking sequence $\pi = (\pi_1 \dots \pi_{i-1}\pi_i \dots \pi_j\pi_{j+1} \dots \pi_n)$ from class A, and applying P on A we have

$$P \cdot [(\pi_1 \dots \pi_{i-1}\pi_i \dots \pi_j\pi_{j+1} \dots \pi_n)] = [(\pi_1 \dots \pi_{i-1}\overline{\pi}_j \dots \overline{\pi}_i\pi_{j+1} \dots \pi_n)]$$

Applying Q on A we have

$$Q \cdot [(\pi_1 \dots \pi_{i-1}\pi_i \dots \pi_j\pi_{j+1} \dots \pi_n)] = [(\overline{\pi}_1\overline{\pi}_n \dots \overline{\pi}_{j+1}\pi_i \dots \pi_j\overline{\pi}_{i-1} \dots \overline{\pi}_2)]$$

But applying r^{-1} and s on $(\pi_1 \dots \pi_{i-1}\overline{\pi}_j \dots \overline{\pi}_i\pi_{j+1} \dots \pi_n)$ we have

$$r^{-1}s \cdot (\pi_1 \dots \pi_{i-1}\overline{\pi}_j \dots \overline{\pi}_i\pi_{j+1} \dots \pi_n) = (\overline{\pi}_1\overline{\pi}_n \dots \overline{\pi}_{j+1}\pi_i \dots \pi_j\overline{\pi}_{i-1} \dots \overline{\pi}_2)$$

$$[\pi] = \{(+1\ -5\ +4\ -3\ +2)\ (-5\ +4\ -3\ +2\ +1)\ (+4\ -3\ +2\ +1\ -5)$$

$$(-3\ +2\ +1\ -5\ +4)\ (+2\ +1\ -5\ +4\ -3)$$

$$(-2\ +3\ -4\ +5\ -1)\ (+3\ -4\ +5\ -1\ -2)\ (-4\ +5\ -1\ -2\ +3)$$

$$(+5\ -1\ -2\ +3\ -4)\ (-1\ -2\ +3\ -4\ +5)\}$$

$$can\ ([\pi]) = (+1\ -5\ +4\ -3\ +2)$$

Figure 2.7: Example of an equivalence class $[\pi]$ and its canonical representative $can([\pi])$.

So

$$[P \cdot A] = [Q \cdot A]$$

■

Now we can enunciate the problem of finding the minimal number of reversals acting on circular chromosomes with known relative orientations.

Given two equivalence classes A and B, representing two circular chromosomes with known relative orientations, the **problem of reversal distance of signed circular chromosomes** is to find a series of reversals $P_1, P_2, \ldots, P_u$ such that $P_u \cdot P_{u-1} \cdot \ldots \cdot P_2 \cdot P_1 \cdot A = B$ and u is minimum. We call u the **reversal distance** of A and B, denoted by $d^c(A, B)$.

2.2.4 Relating Circular Chromosomes to Linear Chromosomes

In the formalization of circular chromosomes, we would like to use some results from the linear case. A linear reversal $\varrho(i, j)$ acts as described in Section 2.2.1. It would be tempting to define a corresponding circular reversal $\varrho^c(i, j)$ by

$$\varrho^c(i, j) \cdot [\pi] = [\varrho(i, j) \cdot \pi]$$

However this definition does not make sense, because different choices of sequences π inside an equivalence class A lead to non-equivalent right-hand members. So, it will not be permitted a random choice of the sequence in A in which the reversal will act.

We will define a *canonical representative* of A, denoted by $can(A)$, with the characteristics of having the 1 block fixed as the first element of the sequence, and with the + orientation (see Figure 2.7).

Note that each equivalence class has a unique canonical representative. For the formalism, a reversal will be applied only in the canonical representative. Thus, given a linear reversal $\varrho(i, j)$ with $1 \leq i \leq j \leq n$, we define a circular reversal $\varrho^c(i, j)$ by the formula

$$\varrho^c(i,j) \cdot A = [\varrho(i,j) \cdot can(A)]$$

Notice that the case $(i,j) = (1,n)$ is excluded from consideration as mentioned in Section 2.2.3.

The next theorem tells us that every circular reversal is of the form $\varrho^c(i,j)$ for some i,j. Moreover, we can always choose the indices from 2 to n.

Theorem 2.2.1 *For any circular reversal P, there are integers i and j with $2 \leq i \leq j \leq n$ such that*

$$P \cdot A = [\varrho(i,j) \cdot can(A)].$$

Proof: A circular reversal P must be applied only in the canonical representative of the equivalence class A representing the circular chromosome. There are two possible forms for a reversal acting on any sequence of A, but both of them produce sequences that belong to the same equivalence class (Lemma 2.2.1). As we can choose any of these forms we will pick the form not including $\pi_1 = +1$. This way $P \cdot A$ will produce a sequence which is also a canonical representative. In other words, the canonical representative of the equivalence class which models the circular chromosome before the reversal is carried to a canonical representative of the equivalence class which represents the circular chromosome after the reversal. In this case, in terms of the linear representation, the reversal acts in the canonical representative like a linear reversal $\varrho(i,j)$. Then,

$$\varrho(i,j) \cdot (+1\ \pi_2 \ldots \pi_i \ldots \pi_j \ldots \pi_n) = (+1\ \pi_2 \ldots \overline{\pi}_j \ldots \overline{\pi}_i \ldots \pi_n)$$

with $2 \leq i \leq j \leq n$. This comes from the definition of linear reversal. As the right sequence is canonical we have

$$\varrho(i,j) \cdot can(A) = can(P \cdot A)$$

from where

$$[\varrho(i,j) \cdot can(A)] = [can(P \cdot A)] = P \cdot A.$$

■

We will see now that there is an isomorphism between reversals acting on circular chromosomes and reversals acting on linear chromosomes. To prove this, we will initially define two bijections. Recall that S_n is the set of all signed linear permutations on n elements. Let R_n be the set of all linear reversals on n elements, and S_n^c, R_n^c the analogous sets for the circular case. Define

$$\varphi : S_n^c \longrightarrow S_{n-1}$$

so that

$$\varphi(A) = \text{ take } can(A), \text{ remove } +1, \text{ subtract 1 from the others}$$

and

$$\theta : R_n^c \longrightarrow R_{n-1}$$

so that

$$\theta(P) = \varrho(i-1, j-1)$$

where $P = \varrho^c(i,j)$, $2 \leq i \leq j \leq n$.

We enunciate the result.

Theorem 2.2.2 *Given the two bijections φ and θ defined above, we have*

$$\varphi(P \cdot A) = \theta(P) \cdot \varphi(A)$$

Proof:

First we have

$$\varphi(P \cdot A) = \text{ take } can(P \cdot A), \text{ remove } +1, \text{ subtract 1 from the others}$$

Let (by Theorem 2.2.1) $P = \varrho^c(i,j)$, with $2 \leq i \leq j \leq n$, and $A = [\pi]$, where $\pi_1 = +1$. Then,

$$\varphi(P \cdot A) = \text{ take } \varrho(i,j) \cdot \pi, \text{ remove } +1, \text{ subtract 1 from the others}$$

On the other side, since $can(A) = \pi$, we have

$$\varphi(A) = \text{ take } \pi, \text{ remove } +1, \text{ subtract 1 from the others}$$

and

$$\theta(P) = \varrho(i-1, j-1)$$

Then we have the result, because $\theta(P)$ will act on the same elements as $\varrho(i,j)$.

■

Note that $\mid S_n^c \mid = \mid S_{n-1} \mid = 2^{n-1}(n-1)!$ and $\mid R_n^c \mid = \mid R_{n-1} \mid = ((n-1)n)/2$.

From Theorem 2.2.2 we have immediately.

Corollary 2.2.1 *Given any two classes A and B modeling two circular chromosomes, and the bijection φ defined above,*

$$d^c(A,B) = d(\varphi(A), \varphi(B))$$

From Corollary 2.2.1 we can derive an algorithm to the problem of signed circular chromosomes. Basically it consists in running any algorithm solving the problem of signed linear chromosomes, taking as inputs two permutations, obtained from applying the bijection φ in the two classes representing the circular chromosomes.

In particular, if we take the KST algorithm, the complexity of the algorithm is $O(n^2)$ (to find out the input sequences costs $O(n)$ and the KST algorithm has complexity $O(n^2)$), where n is the number of genes blocks of the circular chromosomes.

We can also obtain the circular reversals used, just applying the inverse of θ on each step of the algorithm for the linear chromosome. It does not affect the complexity of the above algorithm because it takes $O(1)$.

2.3 Relating circular chromosomes to linear chromosomes of the same size

In the previous section we saw that there is a distance preserving correspondence between circular chromosomes and linear chromosomes of size one unit smaller. Here we will derive similar results for circular and linear chromosomes of the same size.

First of all, we would like to know what is the relation between $d(\pi, \sigma)$ and $d^c([\pi], [\sigma])$ for any π and σ. As we will see in Theorem 2.3.1, $d^c([\pi], [\sigma]) \leq d(\pi, \sigma)$. Before this, we need three technical lemmas.

Lemma 2.3.1 *Given two linear permutations σ and π, such that $\sigma = q \cdot \pi$ where $q = r$ or s, then for every reversal ϱ there is a reversal ϱ' such that $\varrho \cdot \pi \sim \varrho' \cdot \sigma$.*

Proof:

Let $\pi = (\pi_1 \dots \pi_n)$.

We have two possibilities for q.

- Suppose $\sigma = r \cdot \pi = (\pi_2 \pi_3 \dots \pi_n \pi_1)$ and $\varrho = \varrho(i, j)$ so that

$$\varrho(i, j) \cdot \pi = (\pi_1 \dots \pi_{i-1} \overline{\pi}_j \dots \overline{\pi}_i \pi_{j+1} \dots \pi_n)$$

We have three cases.

1. $i = 1, j = n$:

$$\varrho(1, n) \cdot \pi = (\overline{\pi}_n \dots \overline{\pi}_1)$$

In this case $\varrho = s$. Take $\varrho' = s$ also. We have

$$\varrho \cdot \pi = s \cdot \pi \sim \pi \sim \sigma \sim s \cdot \sigma = \varrho' \cdot \sigma$$

2. $i = 1, j < n$: In this case

$$\varrho(1, j) \cdot \pi = (\overline{\pi}_j \dots \overline{\pi}_1 \pi_{j+1} \dots \pi_n).$$

Then:

$$\varrho(j, n-1) \cdot \sigma = (\pi_2 \pi_3 \dots \pi_{j-1} \pi_j \overline{\pi}_n \dots \overline{\pi}_{j+1} \pi_1)$$

$$s \cdot \varrho(j, n-1) \cdot \sigma = (\overline{\pi}_1 \pi_{j+1} \dots \pi_n \overline{\pi}_j \dots \overline{\pi}_3 \overline{\pi}_2)$$

$$r^{n-j+1} \cdot s \cdot \varrho(j, n-1) \cdot \sigma = (\overline{\pi}_j \dots \overline{\pi}_3 \overline{\pi}_2 \overline{\pi}_1 \pi_{j+1} \dots \pi_n)$$

$$\varrho(1, j) \cdot \pi = r^{n-j+1} \cdot s \cdot \varrho(j, n-1) \cdot \sigma$$

Therefore, taking $\varrho' = \varrho(j, n-1)$ we have

$$\varrho \cdot \pi \sim \varrho' \cdot \sigma.$$

3. $i > 1, j \leq n$: In this case

$$\varrho(i,j) \cdot \pi = (\pi_1 \dots \pi_{i-1} \overline{\pi}_j \dots \overline{\pi}_i \pi_{j+1} \dots \pi_n).$$

Then:

$$\varrho(i-1, j-1) \cdot \sigma = (\pi_2 \pi_3 \dots \overline{\pi}_j \dots \overline{\pi}_i \dots \pi_n \pi_1)$$

$$r^{-1} \cdot \varrho(i-1, j-1) \cdot \sigma = (\pi_1 \pi_2 \pi_3 \dots \pi_{i-1} \overline{\pi}_j \dots \overline{\pi}_i \dots \pi_n)$$

$$\varrho(i,j) \cdot \pi = r^{-1} \cdot \varrho(i-1, j-1) \cdot \sigma$$

Therefore, taking $\varrho' = \varrho(i-1, j-1)$ we have

$$\varrho \cdot \pi \sim \varrho' \cdot \sigma.$$

- Suppose $\sigma = s \cdot \pi = (\overline{\pi}_n \dots \overline{\pi}_1)$ and $\varrho = \varrho(i,j)$ so that

$$\varrho(i,j) \cdot \pi = (\pi_1 \dots \pi_{i-1} \overline{\pi}_j \dots \overline{\pi}_i \pi_{j+1} \dots \pi_n)$$

Then:

$$\varrho(n+1-j, n+1-i) \cdot \sigma = (\overline{\pi}_n \dots \overline{\pi}_{j+1} \pi_i \dots \pi_j \overline{\pi}_{i-1} \dots \overline{\pi}_1) \quad .$$

$$s \cdot \varrho(n+1-j, n+1-i) \cdot \sigma = (\pi_1 \dots \pi_{i-1} \overline{\pi}_j \dots \overline{\pi}_i \pi_{j+1} \dots \pi_n)$$

$$\varrho(i,j) \cdot \pi = s \cdot \varrho(n+1-j, n+1-i) \cdot \sigma$$

Therefore, taking $\varrho' = \varrho(n+1-j, n+1-i)$, we have

$$\varrho \cdot \pi \sim \varrho' \cdot \sigma.$$

■

Lemma 2.3.2 *Given two linear permutations π and σ, such that $\pi \sim \sigma$ then for every reversal ϱ there is a reversal ϱ' such that $\varrho \cdot \pi \sim \varrho' \cdot \sigma$.*

Proof:

Take $\sigma = q_v \cdot q_{v-1} \cdot \ldots \cdot q_1 \cdot \pi$, where $v \geq 0$, and $q_i = r$ or s, for $1 \leq i \leq v$.
This proof will be made by induction on v.

- $v = 0$: just make $\varrho' = \varrho$

- $v > 0$: Take

$$\sigma' = q_{v-1} \cdot \ldots \cdot q_1 \cdot \pi$$

Given ϱ, we want to obtain ϱ' such that

$$\varrho \cdot \pi \sim \varrho' \cdot \sigma$$

By the induction hypothesis, we have

$$\varrho \cdot \pi \sim \varrho'' \cdot \sigma'$$

But, $\sigma = q_v \cdot \sigma'$ and then, using Lemma 2.3.1, there is ϱ' such that

$$\varrho'' \cdot \sigma' \sim \varrho' \cdot \sigma$$

Then,

$$\varrho \cdot \pi \sim \varrho' \cdot \sigma$$

■

Lemma 2.3.3 *Given a permutation π and a reversal ϱ, then*

$$[\varrho \cdot \pi] = P \cdot [\pi]$$

where $P = I$, the identity transformation, or P is a circular reversal.

Proof:

Let σ be the canonical representative of $[\pi]$:

$$\pi \sim \sigma = can([\pi])$$

Lemma 2.3.2 says that given ϱ there is ϱ' such that

$$\varrho \cdot \pi \sim \varrho' \cdot \sigma,$$

hence

$$[\varrho \cdot \pi] = [\varrho' \cdot \sigma]$$

But $\varrho' = \varrho(i, j)$ with $1 \leq i \leq j \leq n$. Then we have two cases:

1. $i = j \oplus 1$. Then, $\varrho' = s$ and

$$[\varrho \cdot \pi] = [s \cdot \sigma] = [\sigma] = [\pi],$$

so $P = I$ works in this case.

2. $i \neq j \oplus 1$. Because σ is canonical,

$$[\varrho(i,j) \cdot [\sigma]] = \varrho^c(i,j) \cdot [\pi]$$

hence $P = \varrho^c(i,j)$ works in this case.

■

Now we show that there are fewer reversals in the circular case than in the linear case when both chromosomes have the same size.

Theorem 2.3.1 *Given any two permutations π and σ,*

$$d(\pi,\sigma) \geq d^c([\pi],[\sigma])$$

Proof:

Take $t = d(\pi,\sigma)$. Then,

$$\varrho_t \cdot \varrho_{t-1} \cdot \ldots \cdot \varrho_1 \cdot \pi = \sigma$$

$$[\varrho_t \cdot \varrho_{t-1} \cdot \ldots \cdot \varrho_1 \cdot \pi] = [\sigma]$$

Using Lemma 2.3.3 we have

$$P'_t \cdot P'_{t-1} \cdot \ldots \cdot P'_1 \cdot [\pi] = [\sigma]$$

where P'_i is either a circular reversal or the identity. Then,

$$d^c([\pi],[\sigma]) \leq t = d(\pi,\sigma)$$

■

We note that it is not true that $d(\pi,\sigma) = d^c([\pi],[\sigma])$, for any π and σ. It is enough to take $\pi = (-2\ +3\ +1)$ and $\sigma = (+1\ +2\ +3)$. We have $d^c([\pi],[\sigma]) = 1$ because $d^c([\pi],[\sigma]) = d(can([\pi]), can([\sigma])) = 1$, where $can([\pi]) = (+1\ -2\ +3)$. But $d(\pi,\sigma) = 3$. To make this computation, it is sufficient to construct the breakpoint graphs of π and σ, and use the formula presented by Hannenhalli and Pevzner [64].

Following we demonstrate another theorem that solves the problem of reversal distance for signed circular chromosomes.

Theorem 2.3.2 *Given two circular chromosomes represented by classes A and B we have*

$$d^c(A,B) = d(can(A), can(B))$$

Proof:

First we will show that

$$d^c(A, B) \leq d(can(A), can(B))$$

By Theorem 2.3.1 we know that $d(\pi, \sigma) \geq d^c([\pi], [\sigma])$. In particular, taking $\pi = can(A)$ and $\sigma = can(B)$, we immediately have the result.

Secondly, we will show that

$$d^c(A, B) \geq d(can(A), can(B))$$

To solve the problem of the reversal distance of signed circular chromosome, we use reversals in the interval $[2, n]$, that act always in the canonical representative sequence. Considering the linear chromosome $can(A)$, initially, $\pi_1 = +1$ is in its correct position, and this is not modified throughout the process. Thus, these reversals supply a series of reversals for the linear case too.

■

From Theorem 2.3.2 we can derive another algorithm for the problem of signed circular chromosomes that consists in running any algorithm solving the problem of signed linear chromosomes giving as input the canonical representatives of A and B.

Let us take the two input permutations α and β, where α is a permutation of the A class which represents one of the circular chromosomes, and β a permutation of the B class which represents the other circular chromosome. The canonical representatives are obtained traversing the two permutations α and β finding the position k of the 1 block. If it has sign $+$ we just apply r^{k-1}, and if it has sign $-$ we apply r^{k-n} followed by s.

In particular, if we take the KST algorithm, the complexity of the algorithm is $O(n^2)$ (to find out the canonical representatives costs $O(n)$ and the KST algorithm has complexity $O(n^2)$), where n is the number of gene blocks of the circular chromosomes.

Finally we prove a theorem that allow us to say that the canonical representatives of the classes modeling the circular chromosomes provide a minimum distance, among all permutations belonging to those two classes.

Theorem 2.3.3 *Given any two classes A and B modeling circular chromosomes, we have*

$$d(can(A), can(B)) = \min_{\substack{\pi \in A \\ \sigma \in B}} \{d(\pi, \sigma)\}.$$

Proof:

To begin with, notice that we have $d^c(A, B) = d(can(A), can(B))$ (from Theorem 2.3.2). From Theorem 2.3.1, we have each value $d(\pi, \sigma)$ greater than or equal to $d^c([\pi], [\sigma])$.

■

A question arises here. Which sequences, from the two equivalence classes modeling the circular chromosomes, lead to a minimum reversal distance? Our results showed that the canonical representatives from the classes certainly do. But they are not the only ones. An example found in an article of Palmer and co-authors [102] did not have the characteristics of our canonical representatives, but led to a minimum distance. The sequences in that case were $(-8\ -7\ -6\ -5\ -4\ -3\ -2\ -1\ -11\ -10\ -9C\ -9B\ -9A)$ and $(-4\ +3\ -2\ +8\ +7\ -1\ -5\ -6\ -11\ +10\ +9A\ -9B\ +9C)$. If we call *optimal representatives* of two classes modeling circular chromosomes, two permutations, one for each class, that lead to a minimum reversal distance, we would like to know how to characterize this set of optimal representatives.

From the above results, it can be shown that Corollary 2.2.1 and Theorem 2.3.2 are equivalent, in the following sense:

Theorem 2.3.4 *Given two classes A and B modeling two linear chromosomes and the bijection φ defined above, then*

$$d(\varphi(A), \varphi(B)) = d(can(A), can(B))$$

2.4 The reversal diameter of signed chromosomes

The **circular reversal diameter**, denoted by $D^c(n)$, of the equivalence classes on S_n, with respect to the circular reversal distance, is the maximum distance between two equivalence classes. Similarly, the **linear reversal diameter**, denoted by $D(n)$, of the n element permutations of the set S_n, with respect to the linear reversal distance, is the maximum distance between two permutations. We show now that the reversal diameter for signed linear and circular chromosomes are respectively $n+1$ and n (except in a few cases). This corrects a statement from Kececioglu and Sankoff [77] that said that $n-2 \leq D(n) \leq n-1$.

Now we need some definitions and facts about hurdles and fortresses, as mentioned earlier. A cycle C is **bad** when for any reversal ϱ acting on two reality edges of C we have

$$c(\pi, \sigma) = c(\varrho \cdot \pi, \sigma).$$

Otherwise, the cycle is **good**.

Two cycles are **interleaving** when there are two desire edges, one from each cycle, that cross. A cycle C is **contained** in another cycle D when C and D are not interleaving and C is contained in at least one desire edge of D.

The following facts will be important in this section:

- If a bad cycle C does not interleave with and does not contain any other cycle, then C forms a **hurdle** just by itself. We should point out that these are not the only types of hurdles that can exist in a breakpoint graph, but this will suffice for our purposes.

- In a **fortress** there is at least one cycle that does not belong to a hurdle. Again, we point out that this condition is not sufficient to define fortresses.

Theorem 2.4.1 *The reversal diameter of linear chromosomes is*

$$D(n) = \max_{\substack{\pi \in S_n \\ \sigma \in S_n}} \{d(\pi, \sigma)\} = \begin{cases} n & \text{if } n = 1 \text{ or } n = 3 \\ n+1 & \text{otherwise} \end{cases}$$

Proof:

We will show two sequences, π_n and σ_n, that give $d(\pi_n, \sigma_n) = n + 1$, for each n. The construction depends on n being even or odd. All the integers will have + sign in our examples, so we omit them in the proof.

- For n even, $n \geq 2$, consider

$$\pi_n = (2\ 1\ 4\ 3\ 6\ 5\ \ldots\ n-4\ n-5\ n-2\ n-3\ n\ n-1)$$

and

$$\sigma_n = \iota_n = (1\ 2\ 3\ 4\ \ldots\ n-1\ n)$$

The breakpoint graph for π_n with respect to ι_n is formed by exactly one cycle, of size $n + 1$, involving all labels. This is a bad cycle and therefore a hurdle. Figure 2.8 shows examples of breakpoint graphs for $n = 2$ and 4 with respect to ι_n.

In this case, using the Hannenhalli and Pevzner formula [64], and by construction of the breakpoint graph $G(\pi_n, \sigma_n)$ of the sequence π_n with respect to ι_n,

$$d(\pi_n, \iota_n) = (n+1) - 1 + 1 + 0 = n+1$$

- For n odd, $n = 2 \cdot k + 1$ with $k \geq 0$

Initially we observe that for $n = 1$ we have just two permutations with distance 1 between them, so $D(1) = 1$.

For $n = 3$ we have from a theorem of Kececioglu and Sankoff [77] that the greedy algorithm sorts any permutation π with at least one negative element in at most $n - 1$ steps. Then, only the permutations with all elements positive are candidates to have $D(n) = n + 1$. Using this fact and constructing the breakpoint graphs for all possible sequences with all their elements positive for $n = 3$, we conclude that $d(\pi_3, \iota_3) \leq 3$. On the other hand, $\pi_3 = (3\ 2\ 1)$ and $\iota_3 = (1\ 2\ 3)$ give $d(\pi_3, \iota_3) = 3$, so $D(3) = 3$.

Now we will present sequences π_n such that $d(\pi_n, \iota_n) = n + 1$ for the other cases of n odd, $n = 2 \cdot k + 1$ with $k \geq 2$, that is, $n \geq 5$. We consider now the *remainder* between $n + 1$ and 3. We have three cases:

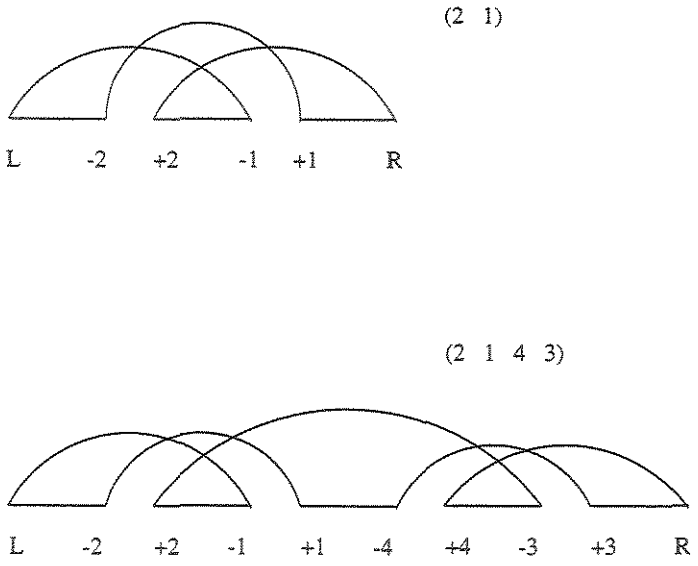

Figure 2.8: The breakpoint graph for $n = 2$ e 4 with respect to ι_n.

- $remainder = 0$: Consider the sequence

$$(2\ 1\ 3\ 5\ 4\ 6\ 8\ 7\ 9\ \cdots\ n-6\ n-7\ n-5\ n-3\ n-4\ n-2\ n\ n-1)$$

The breakpoint graph for π_n with respect to ι_n with $n \geq 5$ is formed by exactly $(n+1)/3$ cycles of size 3, with $n \geq 5$, constructed one beside the other. These are bad cycles and therefore hurdles. Figure 2.9 shows an example of a breakpoint graph for $n = 5$ with respect to ι_5.

In this case, using Hannenhalli and Pevzner formula [64], and by construction of the breakpoint graph $G(\pi_n, \sigma_n)$ of π_n with respect to ι_n,

$$d(\pi_n, \iota_n) = (n+1)\ -\ (n+1)/3\ +\ (n+1)/3\ +\ 0 = n+1$$

- $remainder = 1$: Consider the sequence

$$(2\ 1\ 3\ 5\ 4\ 6\ \cdots\ n-15\ n-13\ n-14\ n-12\ n-10\ n-11$$

$$n-9\ n-7\ n-8\ n-5\ n-6\ n-4\ n-2\ n-3\ n\ n-1)$$

The breakpoint graph for π_n with respect to ι_n with $n \geq 9$ is formed by exactly $(n-9)/3$ cycles of size 3, and 2 cycles of size 5, with $n \geq 9$, constructed one beside the other. These are bad cycles, and therefore hurdles. We note that the restriction $n \geq 9$ does not eliminate any n such that $(n+1) \bmod 3 = 1$, because $n = 9$ is the first odd number satisfying $n \geq 5$. Figure 2.9 shows an example of a breakpoint graph for $n = 9$ with respect to ι_9.

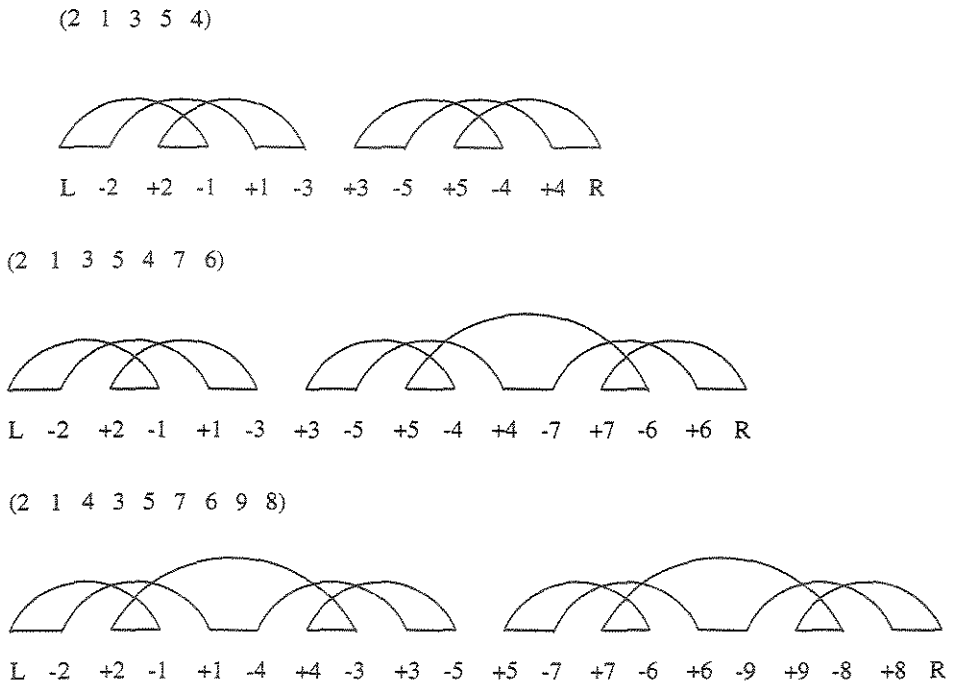

Figure 2.9: The breakpoint graphs for $n = 5, 7$ and 9 with respect to ι_n.

In this case, using the Hannenhalli and Pevzner formula [64], and by construction of the breakpoint graph $G(\pi_n, \sigma_n)$ of π_n with respect to ι_n,

$$d(\pi_n, \iota_n) = (n+1) \; - \; ((n-9)/3+2) \; + \; ((n-9)/3+2) \; + \; 0 = n+1$$

- $remainder = 2$: Consider the sequence

$$(2\,1\,3\,5\,4\,6\,\cdots\,n-11\;n-12\;n-10\;n-8\;n-9\;n-7\;n-5\;n-6$$

$$n-4\;n-2\;n-3\;n\;n-1)$$

The breakpoint graph for π_n with respect to ι_n with $n \geq 7$ is formed by exactly $(n-4)/3$ cycles of size 3, and 1 cycle of size 5, with $n \geq 7$, constructed one beside the other. These are bad cycles, and so hurdles. We note that the restriction $n \geq 7$ does not eliminate any n such that $(n+1) \bmod 3 = 2$, because $n = 7$ is the first odd number satisfying $n \geq 5$. Figure 2.9 shows an example of a breakpoint graph for $n = 7$ with respect to ι_7.

In this case, using the Hannenhalli and Pevner formula [64], and by construction of the breakpoint graph $G(\pi_n, \sigma_n)$ of π_n with respect to ι_n,

$$d(\pi_n, \iota_n) = (n+1) \; - \; ((n-4)/3+1) \; + \; ((n-4)/3+1) \; + \; 0 = n+1$$

Then, we proved that $D(n) \geq n+1$. We yet have to prove that $D(n) < n+2$, to obtain the wanted result.

We have, by the Hannenhalli and Pevner formula [64],

$$d(\pi_n, \iota_n) = (n+1) - c(\pi_n, \iota_n) + h(\pi_n, \iota_n) + f(\pi_n, \iota_n)$$

First, we have $h(\pi_n, \iota_n) \leq c(\pi_n, \iota_n)$, by definition of $h(\pi_n, \iota_n)$. So, if $h(\pi_n, \iota_n) = c(\pi_n, \iota_n)$, then we have $d(\pi_n, \iota_n) \leq (n+1) + 1$, that is, $d(\pi_n, \iota_n) \leq n+2$. But if $f(\pi_n, \iota_n) = 1$, then necessarily $h(\pi_n, \iota_n) < c(\pi_n, \iota_n)$, and then $d(\pi_n, \iota_n) < n+2$.

This proves the linear case.

■

From the bijections defined earlier, we have the following result.

Lemma 2.4.1

$$D^c(n) = D(n-1)$$

From this lemma, we have the following theorem showing the circular reversal diameter of the equivalence classes on S_n.

Theorem 2.4.2 *The reversal diameter of circular chromosomes is*

$$D^c(n) = \max_{\substack{A \in S_n^c \\ B \in S_n^c}} \{d^c(A,B)\} = \begin{cases} n-1 & \text{if } n = 1,\ n = 2 \text{ or } n = 4 \\ n & \text{otherwise} \end{cases}$$

2.5 Conclusions

In this work, we attempted to start a systematic study of the theory of the reversal distance problem for signed circular chromosomes. To do this, we gave some contributions, described as follows. First we formalized circular chromosomes by equivalence classes. This is interesting because it includes the different forms to visualize a signed circular chromosome, obtained by rotations and reflections. We also defined circular reversals using the known definitions of linear reversals, which allowed to solve the reversal distance problem of signed circular chromosomes by using polinomial algorithms that solve the reversal distance problem of signed linear chromosome, giving as input suitable sequences from the equivalence classes. Besides, we presented some results concerning the linear and circular chromosomes of the same size. Finally, we determined the signed reversal diameter for linear ($D(n) = n+1$) and circular chromosomes ($D^c(n) = n$), correcting a result of Kececioglu and Sankoff [77] on the linear reversal diameter $D(n)$.

To finish, a question arising from these studies is which permutations from the equivalence classes lead to a minimum reversal distance, that is, we would like to know how to characterize precisely the set of optimal representatives.

Capítulo 3

Transposition Distance Between a Permutation and its Reverse *

João Meidanis

Maria Emilia M. T. Walter

Zanoni Dias

Abstract

In this note we solve an open question posed by Bafna and Pevzner [9], regarding chromosome distance with respect to transpositions: we show that the distance between a permutation and its reverse (without complementation) is $\lfloor n/2 \rfloor + 1$, where n is the size of the permutations. We also present an algorithm to compute an optimal series of transpositions.

Trabalho apresentado no **IV South American Workshop on String Processing** *(WSP'97), realizado na cidade de Valparaíso, no Chile, em novembro de 1997.*

3.1 Introduction

The huge amount of data resulting from genome sequencing in Molecular Biology is giving rise to an increasing interest in the development of algorithms for comparing genomes of related species. Particularly these data prompted research on mutational events acting on large portions of the chromosomes. Such events can be used to compare genomes for which the traditional methods of comparing DNA sequences are not conclusive. The field originated by the study of large mutations on chromosomes is known as *genome rearrangements*.

There are several mutational events affecting large fragments of genomes of organisms, including duplication, reversal, transposition (acting on a single chromosome), translocation, fusion, and fission (involving more than one chromosome). Each such event or combination of events gives rise to a theoretical problem of finding, given two genomes, the shortest series of events that transforms one genome into the other. We seek the shortest series because it has the largest likelihood of occurrence under a general principle of parsimony. Notice that in general more than one shortest series exists. The length of the shortest series is called the *distance* between the two genomes.

Chromosomes are usually represented as *permutations* of integers in a given range, each integer representing a gene. Sometimes the integers are signed to indicate the orientation of the gene. However, when the gene orientations are unknown or not relevant (as in the case of transpositions), the integers are unsigned.

In the last few years we have witnessed formidable advances in our understanding of genome rearrangements. A partial list of known results follows. With respect to the *reversal* event, Hannenhalli and Pevzner [64] presented the first polynomial time algorithm to find the distance, later improved on its running time by Berman and Hannenhalli [13], and Kaplan, Shamir, and Tarjan [73]. These results concern signed permutations. For the unsigned case, also involving reversals, Caprara [18, 21] showed that finding the distance is NP-hard. Hannenhalli and Pevzner [62] studied a multichromosomal distance problem for signed genomes involving reversals, fusion, fission, and a specific form of translocation, producing a polynomial time algorithm in this case as well. Bafna and Pevzner [9] analyzed the problem with respect to transpositions, presenting several approximation algorithms, and leaving a number of open questions, among them the complexity of the problem and the diameter (largest possible distance between two permutations of size n). Gu, Peng, and Sudborough [56, 55] gave approximation algorithms for the combination of events of reversal and transposition.

In this work we solve an open question posed by Bafna and Pevzner [9], regarding chromosome distance with respect to transpositions: we show that the distance between a permutation and its reverse (without complementation) is $\lfloor n/2 \rfloor + 1$, where n is the size of the permutations. Besides, we present an algorithm to compute an optimal series of transpositions.

3.2 Definitions

Chromosomes are represented by *permutations* of integers in the range $1..n$, where n is the number of genes of interest in the chromosome. For instance, (3 4 2 6 1 5) represents a chromosome with six genes. A *transposition* is an operation that transforms a permutation into another one, "cutting" a certain portion of the permutation and "pasting" it elsewhere in the same permutation. A transposition $\rho(i, j, k)$ is defined by three integers i, j, and k such that $1 \leq i < j \leq n + 1$, $1 \leq k \leq n + 1$, and $k \notin [i, j]$, in the following way. It "cuts" the portion between positions i and $j - 1$, including the extremes, and "pastes" it just before position k. Thus, we can write

$$\rho(i, j, k) \cdot (\pi_1 \pi_2 \dots \pi_i \dots \pi_j \dots \pi_k \dots \pi_n) = \\ (\pi_1 \pi_2 \dots \pi_{i-1} \pi_j \dots \pi_{k-1} \pi_i \dots \pi_{j-1} \pi_k \dots \pi_n),$$

if $i < j < k$, and

$$\rho(i, j, k) \cdot (\pi_1 \pi_2 \dots \pi_k \dots \pi_i \dots \pi_j \dots \pi_n) = \\ (\pi_1 \pi_2 \dots \pi_{k-1} \pi_i \dots \pi_{j-1} \pi_k \dots \pi_{i-1} \pi_j \dots \pi_n),$$

if $k < i < j$. Notice that $\rho(i, j, k) = \rho(j, k, i)$ when $i < j < k$.

Given two permutations π and σ, the *transposition distance* or just *distance* between them is the minimum number t of transpositions $\varrho_1 \dots \varrho_t$ such that

$$\varrho_t \varrho_{t-1} \dots \varrho_1 \cdot \pi = \sigma.$$

We denote such distance by $d(\pi, \sigma)$. Because the inverse of a transposition is also a transposition, we have that $d(\pi, \sigma) = d(\sigma, \pi)$.

A powerful tool for studying the transposition distance is the *reality and desire diagram* of two permutations. Suppose we want to compute $d(\pi, \sigma)$. We construct this diagram writing the origin permutation π in the following way. Replace each integer i by a pair of points $-i$ and $+i$, in this order, and add two extra points, one called $+0$ at the beginning of the sequence, and one called $-(n + 1)$ at the end of the sequence. Now draw oriented *reality* edges from $-\pi_1$ to $+0$, from $-\pi_{i+1}$ to $+\pi_i$, and from $-(n + 1)$ to $+\pi_n$. Finally, draw oriented *desire* edges from $+0$ to $-\sigma_1$, from $+\sigma_i$ to $-\sigma_{i+1}$, and from $+\sigma_n$ to $-(n + 1)$.

The diagram has exactly $n + 1$ reality edges and the same number of desire edges. The idea is that reality edges indicate the situation as it is now, while desire edges indicate the situation sought. When reality equals desire in all edges, we have $\pi = \sigma$ and $d = 0$. Therefore, in a way, our goal is to apply transpositions so that reality becomes closer to desire. Figure 3.1 shows the diagram corresponding to a pair of permutations.

Bafna and Pevzner [9] made several useful results regarding the reality and desire diagram. One of them is that the diagram is composed of a number of cycles, with each cycle alternating

π = (8 5 1 4 3 2 7 6)

σ = (1 2 3 4 5 6 7 8)

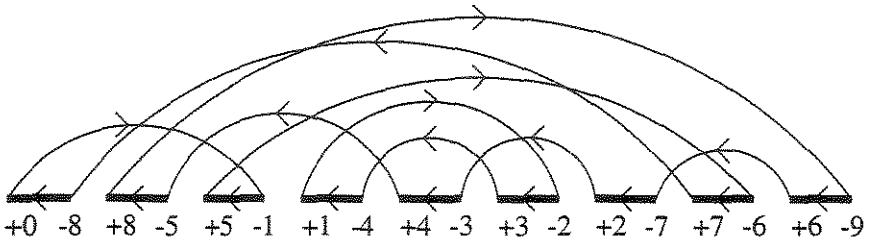

Figure 3.1: Reality and desire diagram for two permutations, π and σ, as showed. In this figure, reality edges are represented by thick lines and desire edges by thin lines.

between reality and desire edges. The *length* of a cycle is the number of reality edges in it (which is the same as the number of desire edges in it). One important remark follows.

Lemma 3.2.1 *The sum of the lengths of all cycles in any reality and desire diagram is always equal to* $n + 1$.

Moreover, a transposition can affect the number of cycles in a very specific way, as the following lemma shows [9]. Denote $c(\pi, \sigma)$ the number of cycles in the diagram of π and σ.

Lemma 3.2.2 *For any permutations* π *and* σ *and any transposition* ϱ *we have*

$$c(\varrho \cdot \pi, \sigma) = c(\pi, \sigma) + x,$$

where $x = -2$, *0, or 2.*

A transposition ϱ is called a -2*-move*, a *0-move*, or a *2-move* according to x being -2, 0, or 2 in the previous lemma. Since $c(\sigma, \sigma) = n + 1$, the maximum possible, we would like to perform 2-moves as much as possible.

If fact, a stronger statement can be made regarding the effect of a transposition on a diagram. Denote by $c_{odd}(\pi, \sigma)$ the number of cycles of odd length in the diagram of π and σ.

Lemma 3.2.3 *For any permutations* π *and* σ *and any transposition* ϱ *we have*

$$c_{odd}(\varrho \cdot \pi, \sigma) = c_{odd}(\pi, \sigma) + x,$$

where $x = -2$, *0, or 2.*

From this lemma we have the following lower bound on the distance:

Theorem 3.2.1 *For any permutations π and σ we have*

$$d(\pi, \sigma) \geq \frac{(n+1) - c_{odd}}{2}$$

Now we make some definitions used in the following sections.

First we show a way to represent a cycle by its reality edges. We number the reality edges of the diagram assigning label i to a reality edge from π_{i+1} to π_i, with $0 \leq i \leq n$, so we label them from 1 to $n+1$. Let us consider a cycle C of size k, taking the reality edges in the order they appear in the cycle, $(i_1, \ldots, i_k)$. A cycle C can be represented in k possible ways, depending on the choice of the first reality edge. We will consider a *canonical representative of a cycle C*, taking the initial reality edge i_1 as the rightmost edge of C in π, that is, $i_1 = \max_{1 \leq t \leq k} i_t$. In the diagram of Figure 3.1 we have three cycles, with canonical representatives $C_1 = (9, 7, 5, 2)$, $C_2 = (8, 1, 3)$ and $C_3 = (6, 4)$.

Let us consider now three reality edges x, y, z belonging to the same cycle C in the diagram. C forces an order on x, y, z, and we have three possible representations of this order. We will choose as the *canonical representative of a triple* (x, y, z) the one starting from the rightmost reality edge $\max(x, y, z)$. A triple in the canonical order is *non-oriented* if $x > y > z$ and *oriented* if $y < z < x$. In the diagram of Figure 3.1 we have the following non-oriented triples: $(9, 7, 5)$, $(9, 7, 2)$, $(9, 5, 2)$, and $(7, 5, 2)$; and the oriented triple $(8, 1, 3)$.

Finally, we say that a cycle is *oriented* if it admits a 2-move, and *non-oriented* if there is no possible 2-moves acting on it. In the diagram of Figure 3.1 we have C_1 and C_3 non-oriented and C_2 oriented.

3.3 Computing the transposition distance

Given the permutations $\pi = (n\ n-1\ n-2\ \ldots\ 2\ 1)$ and $\iota = (1\ 2\ \ldots\ n-2\ n-1\ n)$ we want to compute the transposition distance $d(\pi, \iota)$. Of course, this distance will be the same for any pair consisting of a permutation and its reverse. Theorem 3.3.1 below establishes that $d(\pi, \iota) = 1$ if $n = 2$ and $d(\pi, \iota) = \left\lfloor \frac{n}{2} \right\rfloor + 1$ if $n > 2$.

However, in the proof of this theorem we need the following lemma, which can be easily proved. Bafna and Pevzner [9] mention part of this result in their work.

Lemma 3.3.1 *Let C be a cycle and (x, y, z) a triple of C in the canonical representation. Then we have*

$$(x, y, z) \text{ is oriented if and only if } \varrho(y, z, x) \text{ is a 2-move}$$

and

$$(x, y, z) \text{ is non-oriented if and only if } \varrho(y, z, x) \text{ is a 0-move}$$

Now we will state and prove the main theorem.

Theorem 3.3.1 *Given the permutations* $\pi = (n\ n-1\ n-2\ \ldots\ 2\ 1)$ *and* $\iota = (1\ 2\ \ldots\ n-2\ n-1\ n)$, *we have for* $n \geq 2$

$$d(\pi, \iota) = \begin{cases} 1 & \text{if } n = 2 \\ \\ \left\lfloor \frac{n}{2} \right\rfloor + 1 & \text{if } n > 2 \end{cases}$$

Proof: From the work of Bafna and Pevzner [9], given $\pi = (n\ n-1\ n-2\ \ldots\ 2\ 1)$ we have

$$c_{odd}(\pi, \iota) = \begin{cases} 1 & \text{if } n \text{ is even} \\ \\ 0 \text{ ou } 2 & \text{if } n \text{ is odd} \end{cases}$$

Applying the lower bound given by Theorem 3.2.1 for $d(\pi, \iota)$ we have

$$d(\pi, \iota) \geq \frac{(n+1) - c_{odd}}{2} = \begin{cases} \frac{(n+1)-1}{2} = \frac{n}{2} & \text{if } n \text{ is even} \\ \\ \frac{n+1}{2} \text{ ou } \frac{n-1}{2} & \text{if } n \text{ is odd} \end{cases}$$

Notice that in order to attain the lower bound every transposition used must increase the number of odd cycles.

For π and ι as defined earlier in this section Bafna and Pevzner [9] proved the following upper bound

$$d(\pi, \iota) \leq \left\lfloor \frac{n}{2} \right\rfloor + 1$$

for all $n \geq 1$.

We have two cases:

1. When n is odd:

   - For the case of 0 odd cycles:

     $$d(\pi, \iota) \geq \frac{n+1}{2} = \left\lfloor \frac{n}{2} \right\rfloor + 1$$

     In that case, the lower bound is exactly equal to the upper bound, and then $d(\pi, \iota) = \left\lfloor \frac{n}{2} \right\rfloor + 1$.

   - For the case of 2 odd cycles:

     $$d(\pi, \iota) \geq \frac{n-1}{2} \neq \left\lfloor \frac{n}{2} \right\rfloor + 1$$

     and the gap is exactly 1. But here we have two non-oriented odd cycles. This implies that the next move cannot increase c_{odd}, and therefore we cannot reach the lower bound.

So, when n is odd then we have $d(n\ n-1\ n-2\ \ldots\ 2\ 1) = \left\lfloor \frac{n}{2} \right\rfloor + 1$.

2. When n is even:

$$d(\pi, \iota) \geq \frac{n}{2} \neq \left\lfloor \frac{n}{2} \right\rfloor + 1$$

and the gap is exactly 1. In this case, we have to prove that there is necessarily a transposition that will not increase c_{odd} during any transposition sequence that transforms π into ι.

The first transposition is either a 0-move or a 2-move. We cannot apply a -2-move because this first diagram is formed by just one cycle.

If we apply a 0-move, the unique odd cycle is transformed into another odd cycle, not increasing c_{odd}.

So, we have to verify what happens if we apply a 2-move. We will show that any 2-move gives rise to a diagram with all cycles non-oriented. This will imply the result as follows. We have two possibilities. If the resulting diagram has one odd cycle and two even cycles, the first transposition did not increase c_{odd}. On the other hand, if we end up with three odd cycles, the second transposition of the series cannot increase c_{odd}, because all three cycles are non-oriented.

Let us now study what happens when the first transposition is a 2-move. From Lemma 3.3.1 we know that every 2-move corresponds to an oriented triple in the unique cycle of the diagram. The order of reality edges in this cycle is such that all even labels appear together, in decreasing order, and all odd labels appear together, also in decreasing order. It follows that the possible 2-moves are of the form $\varrho(i, j, k)$ with i and j of opposite parity, k of same parity as i, and $1 \leq i < j < k \leq n+1$.

We now have to apply one such transposition and analyze the resulting diagram. Because $i < j < k$, we have

$$\begin{aligned}\varrho(i,j,k) \cdot (n\ n-1\ n-2\ \ldots\ 2\ 1) = \\ (\ n-1\ \ldots\ n-i+2 \quad n-j+1\ \ldots\ n-k+2 \\ n-i+1\ \ldots\ n-j+2 \quad n-k+1\ \ldots\ 2\ 1).\end{aligned}$$

Figure 3.2 shows examples of such 2-moves.

The important fact here is that, because of the opposite parity of j and k, and of i and j, the cycle involving reality edge $(n-i+1, n-k+2)$ is non-oriented. Likewise, the cycle involving reality edge $(n-k+1, n-j+2)$ is non-oriented because of the opposite parity of i and j (regardless of the parity of k), and the cycle involving reality edge $(n-j+1, n-i+2)$ is also non-oriented because j and k have opposite parity (regardless

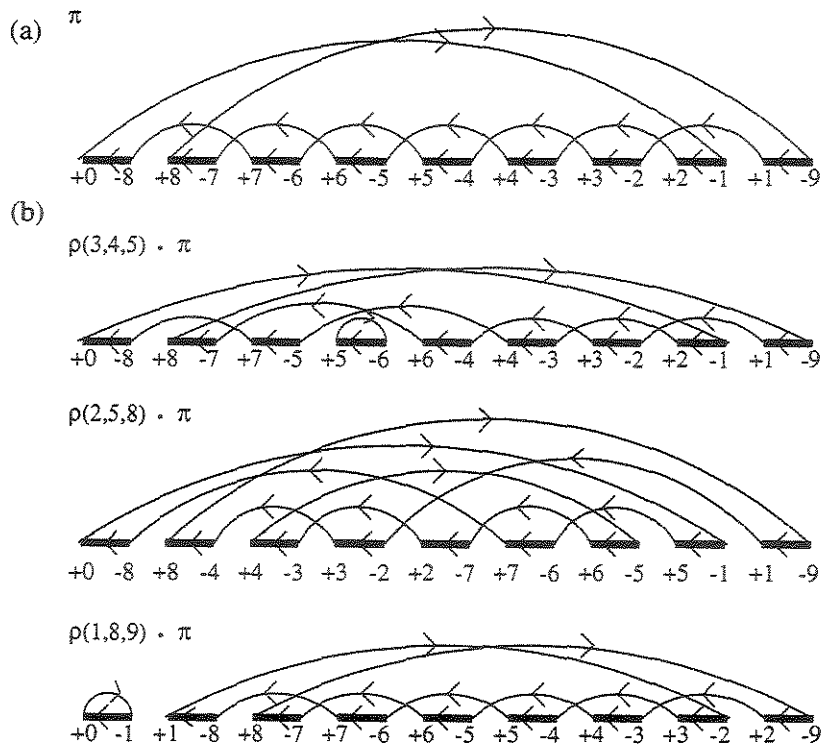

Figure 3.2: This figure shows the diagram created by a strictly decreasing sequence with respect to the identity, and the diagrams created by some possible 2-moves applied to the first diagram. (a) The diagram created by the decreasing cycle $\pi = (8\ 7\ 6\ 5\ 4\ 3\ 2\ 1)$ with respect to $\tau = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8)$. (b) The diagrams created by some transpositions applied to π as indicated.

of the parity of i). Thus, in any case we end up with three non-oriented cycles, which proves the theorem.

■

3.4 An algorithm to compute $d(\pi, \tau)$

We show now an algorithm to compute the transposition distance between a strictly decreasing sequence with respect to the identity. Note that the algorithm runs without using the reality and desire diagram. Instead, it uses an explicit series of transpositions that work in the case treated in this article. As the series has length $\lfloor n/2 \rfloor + 1$, the results in the previous section guarantee that it is a shortest series.

Algorithm

Input: $n > 2, \pi = (n\ n-1\ \ldots\ 2\ 1)$

Output: $t = d(\pi, \tau)$ and $\varrho_1, \varrho_2, \ldots, \varrho_t$

begin

1. $\pi_1 \leftarrow \varrho_1(1, \lceil \frac{n}{2} \rceil, n) \cdot \pi$
2. $t \leftarrow 1$
3. if n is even then
   $t \leftarrow t + 1$
   $\pi_2 \leftarrow \varrho_t(\frac{n}{2}, \frac{n}{2} + 1, n + 1) \cdot \pi_1$
   $k \leftarrow 1$
   $p \leftarrow 1$
4. if n is odd then
   $k \leftarrow 0$
   $p \leftarrow 0$
5. while $k < \lfloor \frac{n}{2} \rfloor$ do
   $t \leftarrow t + 1$
   $\pi_t \leftarrow \varrho_t(\lfloor \frac{n}{2} \rfloor - k, \lfloor \frac{n}{2} \rfloor - k + 2, n + 1 - k + p) \cdot \pi_{t-1}$
   $k \leftarrow k + 1$
6. return $t, \varrho_1, \varrho_2, \ldots, \varrho_t$

end

The four initial steps create, from the initial permutation, a new permutation with two decreasing subsequences on its left extremity, and an increasing sequence, on its right end. If n is even then we have

$$\pi_2 = (\ \frac{n}{2}+1\ \ \frac{n}{2}\ \ldots\ 3\ )\ (\ n\ n-1\ \ldots\ \frac{n}{2}+2\ )\ (\ 1\ 2\ )$$

Note that the decreasing subsequences have $\lceil \frac{n}{2} \rceil - 1$ elements each. We marked the subsequences with parenthesis.

If n is odd then we have

$$\pi_1 = (\ \frac{n+1}{2}\ \ldots\ 3\ 2\ )\ (\ n\ n-1\ \ldots\ \frac{n+1}{2}+1\ )\ (\ 1\ )$$

Analogously, in this case the first two subsequences also have $\lceil \frac{n}{2} \rceil - 1$ elements each.

The loop in step 5 moves both the last element of first subsequence and the first element of second subsequence to the right end of the permutation, where two other subsequences are being increased as the algorithm runs. Generically, if n is even then we have, after $k-1$ iterations of the loop,

$$\pi_{k+1} = (\frac{n}{2}+1 \;\; \frac{n}{2} \;\ldots\; k+2)\;(n-k+1 \;\ldots\; \frac{n}{2}+2)\;(1\;2\;\ldots\;k+1)\;(n-k+2\;\ldots\;n).$$

If n is odd then we have, after k iterations,

$$\pi_{k+1} = (\frac{n+1}{2} \;\ldots\; k+2)\;(n-k \;\ldots\; \frac{n+1}{2}+1)\;(1\;2\;\ldots\;k+1)\;(n-k+1\;\ldots\;n).$$

So this algorithm correctly transforms the permutation in its inverse, using transpositions. Also, the algorithm runs in $\left\lfloor \frac{n}{2} \right\rfloor + 1$ steps, for $n > 2$.

Figure 3.3 shows examples of this algorithm executions for the decreasing sequences for $n = 6$ and $n = 7$, with respect to the identity.

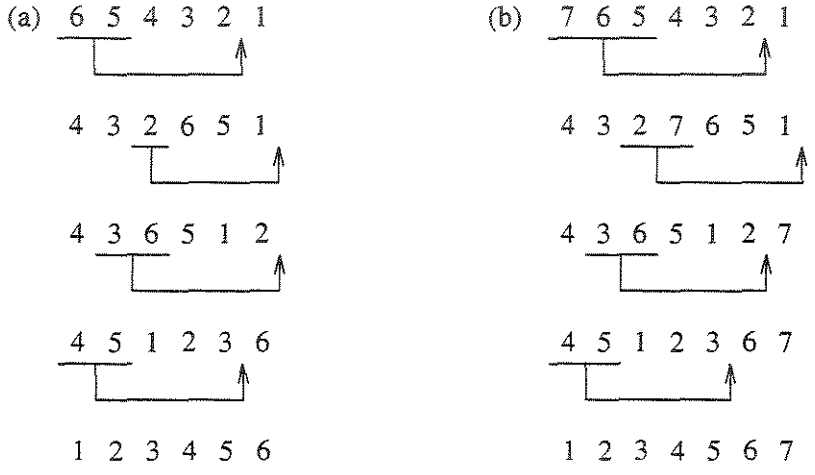

Figure 3.3: This figure shows two executions of the algorithm. (a) Example with n even. (b) Example with n odd.

3.5 Conclusions

We demonstrated that the transposition distance between a permutation and its reverse (without complementation) is $\left\lfloor \frac{n}{2} \right\rfloor + 1$ for all $n > 2$, where n is the size of the permutation. We conjecture that this is in fact the value of the transposition diameter.

We also presented an algorithm to find an optimal series of sorting transpositions for the case studied.

Capítulo 4

Reversal and Transposition Distance of Linear Chromosomes *

Maria Emilia M. T. Walter

Zanoni Dias

João Meidanis

Abstract

In recent years we are seeing increasing interest in research on mutational events acting on large portions of the chromosomes. Among these events, a *reversal* acts on a fragment of a chromosome reversing the order and orientation of the genes, and a *transposition* moves fragments from one region to another within a chromosome. In this article we analyze genomes evolving by reversals and transpositions. We present approximation algorithms to compute the reversal and transposition distance for linear permutations, and a lower bound on the reversal and transposition diameter of signed linear permutations.

* *Trabalho apresentado no* **String Processing and Information Retrieval** *(SPIRE'98) realizado na cidade de Santa Cruz de la Sierra, na Bolívia, em setembro de 1998.*

4.1 Introduction

The huge amount of data resulting from genome sequencing in molecular biology is giving rise to an increasing interest in the development of algorithms for comparing genomes of related species. Particularly these data prompted research on mutational events acting on large portions of the chromosomes. Such events can be used to compare genomes for which the traditional alignment methods of comparing DNA sequences are not conclusive. The field originated by the study of non-local mutations on chromosomes is known as *genome rearrangements.*

There are several mutational events affecting large fragments of genomes of organisms, including duplication, insertion, deletion, reversal, transposition (acting on a single chromosome), translocation, fusion and fission (involving more than one chromosome). Each such event or combination of events gives rise to a theoretical problem of finding, given two genomes, the shortest series of events that transforms one genome into the other. We seek the shortest series because it has the largest likelihood of occurrence under a general principle of parsimony. Notice that in general more than one shortest series exists. The length of the shortest series is called the *distance* between the two genomes.

In this article we are working with genomes composed by a single chromosome. Chromosomes are usually represented as *permutations* of integers in the range $1..n$, for a given n, each integer representing a gene or a genetic marker. Sometimes the integers are signed to indicate the orientation of the gene. However, when gene orientations are unknown, the integers are unsigned.

In the last few years we have witnessed formidable advances in our understanding of genome rearrangements. A partial list of known results follows. With respect to the reversal event, Kececioglu and Sankoff [78] presented the first algorithms for computing the reversal distance between two unsigned linear chromosomes. Bafna and Pevzner [8] improved the Kececioglu and Sankoff algorithm, for signed and unsigned linear permutations. Hannenhalli and Pevzner [64] presented the first polynomial time algorithm to find the reversal distance of signed linear chromosomes, later improved on its running time by Berman and Hannenhalli [13] and Kaplan, Shamir and Tarjan [73]. Caprara, Lancia, and Ng [23] implemented a branch-and-bound algorithm for computing the exact reversal distance between two unsigned permutations which performs very well in practice. Caprara [18, 21] later showed that this problem is NP-hard.

Regarding the problem of reversal distance between two signed circular permutations, Kececioglu and Sankoff [77] gave an approximation algorithm, and Meidanis, Walter and Dias [92, 95] gave a polynomial time algorithm for it. With respect to the transposition event, Bafna and Pevzner [9] analyzed the transposition distance problem between two unsigned linear chromosomes, presenting several approximation algorithms. Christie [28] gave a polynomial time algorithm for computing distance under a novel operation, block interchange.

Analyzing genomes evolving due to different mutational events represents today a great

challenge. Hannenhalli and co-authors [61] analyzed genomes evolving by different events, particularly reversals and transpositions. Hannenhalli and Pevzner [62] presented a polynomial time algorithm for comparing genomes evolving by reversals, translocations, fusions and fissions. Gu, Peng and Sudborough [56, 55] gave approximation algorithms to compute the distance between two signed permutations, allowing three operations, reversal, transposition and reversal+transposition simultaneously.

In this paper we want to contribute in the analysis of reversals and transpositions acting on a single chromosome. The results of this work are as follows. We extend the analysis of transpositions to signed permutations, and obtain approximation algorithms for computing the reversal and transposition distance for both signed and unsigned permutations. Finally, we present lower bound for the reversal and transposition diameter of signed permutations, and conclude.

4.2 Definitions

In this section we formalize the problem of computing the reversal and transposition distance of linear chromosomes.

We assume that the order of genes in a chromosome is represented by a permutation $\pi = (\pi_1, \pi_2 \dots \pi_n)$, where each π_i is an integer in $1..n$. If the gene orientations are known, each π_i is a signed integer.

A *reversal* is an operation that transforms a permutation into another, reversing the order of the genes on a certain portion of the permutation. A reversal $r(i, j)$ is defined by two integers i, j, such that $1 \leq i \leq j \leq n$, reversing the order of the genes between i and j, including the extremes. Thus, we have

$$r(i,j) \cdot (\pi_1 \dots \pi_{i-1}\pi_i\pi_{i+1} \dots \pi_j\pi_{j+1} \dots \pi_n) =$$

$$(\pi_1 \dots \pi_{i-1}\overline{\pi}_j \dots \overline{\pi}_{i+1}\overline{\pi}_i\pi_{j+1} \dots \pi_n)$$

where $\overline{\pi}_k$ symbol means $-\pi_k$ if the integer is signed, or π_k if the integer is not signed.

A *transposition* is an operation transforming a permutation into another, "cutting" a certain portion of the permutation and "pasting" it elsewhere in the same permutation. A transposition $t(i, j, k)$ is defined by three integers i, j, and k such that $1 \leq i < j \leq n+1, 1 \leq k \leq n+1$, and $k \notin [i, j]$, in the following way. It "cuts" the portion between positions i and $j-1$, including the extremes, and "pastes" it just before position k. Thus, we can write

$$t(i,j,k) \cdot (\pi_1 \dots \pi_{i-1}\pi_i \dots \pi_{j-1}\pi_j \dots \pi_{k-1}\pi_k \dots \pi_n)$$

$$(\pi_1 \dots \pi_{i-1}\pi_j \dots \pi_{k-1}\pi_i \dots \pi_{j-1}\pi_k \dots \pi_n)$$

if $i < j < k$, and

$$t(i,j,k) \cdot (\pi_1 \dots \pi_{k-1}\pi_k \dots \pi_{i-1}\pi_i \dots \pi_{j-1}\pi_j \dots \pi_n)$$

$$(\pi_1 \dots \pi_{k-1}\pi_i \dots \pi_{j-1}\pi_k \dots \pi_{i-1}\pi_j \dots \pi_n)$$

if $k < i < j$. Notice that $t(i,j,k) = t(j,k,i)$ when $i < j < k$.

Given two permutations π and σ, we want to compute a shortest series of reversals and transpositions that transforms π into σ, that is, we want to find $\rho_1, \rho_2, \dots, \rho_u$, where ρ_i is either a reversal or a transposition, such that $\rho_u \cdot \rho_{u-1} \cdot \dots \cdot \rho_2 \cdot \rho_1 \cdot \pi = \sigma$ and u is minimum. We call u the **reversal and transposition distance** and denote it by $d(\pi, \sigma)$. Without loss of generality we can fix σ. Unless otherwise noted, all our developments will be done with σ being the identity permutation, which is $\sigma = (1 \dots n)$ in the unsigned case and $\sigma = (+1 \dots +n)$ in the signed case.

In the following an **operation** can be a reversal or a transposition.

We usually extend permutation π by adding $\pi_0 = 0$ and $\pi_{n+1} = n+1$ in the unsigned case, or $\pi_0 = +0$ and $\pi_{n+1} = +(n+1)$ in the signed case. The extended permutation will still be denoted by π.

A **breakpoint** of a permutation π is a pair $x = (\pi_i, \pi_{i+1})$ such that neither x nor $\overline{x} = (\overline{\pi}_{i+1}, \overline{\pi}_i)$ are of the form (σ_j, σ_{j+1}) for some j such that $0 \leq j \leq n$. Therefore, to reach σ from π, we must have at least one operation "separating" π_i and π_{i+1}. Breakpoints are indicated by a bullet ($\bullet$) between π_i and π_{i+1} (see Figure 4.1). We denote by $b(\pi, \sigma)$ the number of breakpoints of π with respect to σ.

Breakpoints divide a permutation into **strips**. If the target permutation σ is the identity, strips are always sequences of consecutive integers. In the unsigned case, a strip can be either increasing or decreasing as a sequence of integers, and we will call them accordingly as **increasing strips** or **decreasing strips**. In the signed case, all strips are increasing, but we separate them into **positive** or **negative** strips, according to the sign of their elements (all elements in a strip must have the same sign).

0 • 5 • 1 2 • 4 • 7 6 • 3 • 9 8 • 10

Figure 4.1: Strips and breakpoints of a permutation π = (0 5 1 2 4 7 6 3 9 8 10) with respect to σ = (0 1 2 3 4 5 6 7 8 9 10). Strips are the sequences between two consecutive breakpoints.

A powerful tool for studying the reversal and transposition distance is the **reality and desire diagram** of two permutations. In the literature [8, 64, 13] this is called the *breakpoint graph* of

two permutations, but we prefer to call it a diagram because its graph structure alone does not capture all the important information: the order of nodes is relevant too.

The rest of this section refers to signed permutations only. We construct this diagram writing the original permutation π in the following way. Replace each integer i by a pair of points $-i$ and $+i$, in this order. For instance, $+4$ is replaced by -4 and $+4$; -8 is replaced by $+8$ and -8. Add two extra points, one called $+0$ at the beginning of the sequence, and one called $-(n+1)$ at the end of the sequence. Now draw **reality** edges between $+0$ and $-\pi_1$, between $+\pi_i$ and $-\pi_{i+1}$, and between $+\pi_n$ and $-(n+1)$. Finally, draw **desire** edges between $+0$ and $-\sigma_1$, between $+\sigma_i$ and $-\sigma_{i+1}$, and between $+\sigma_n$ and $-(n+1)$. Again, in the literature, reality edges are called *black edges* and desire edges are called *gray edges*. We prefer the denominations reality and desire because they are more informative: reality edges refer to the current permutation and desire edges refer to the target permutation.

The diagram has exactly $n+1$ reality edges and the same number of desire edges. The idea is that reality edges indicate the situation as it is now, while desire edges indicate the situation sought. When reality equals desire in all edges, we have $\pi = \sigma$ and $d = 0$. Therefore, our goal is to apply reversals and transpositions so that reality becomes desire. Figure 4.2 shows the diagram corresponding to a pair of permutations. We denote by $G(\pi, \sigma)$ the diagram of the permutations π and σ.

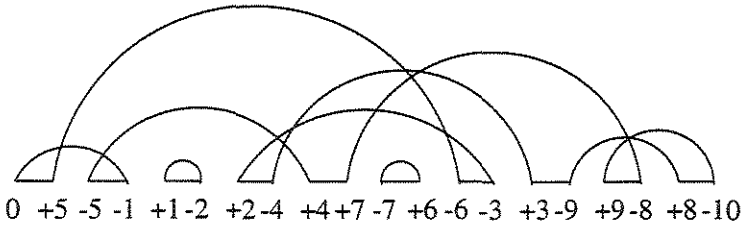

Figure 4.2: Reality and desire diagram for two permutations, $\pi = (-5\ +1\ +2\ +4\ -7\ -6\ +3\ +9\ +8)$ and $\sigma = (+1\ +2\ +3\ +4\ +5\ +6\ +7\ +8\ +9\ +10)$. The value of $c(\pi, \sigma)$ is 3 in this case.

Observe that the diagram is composed of a number of cycles, with each cycle alternating between reality and desire edges. The **length** of a cycle is the number of reality edges in it (which is the same as the number of desire edges in it). We will denote by **k-cycle** a cycle with length k. The decomposition of $G(\pi, \sigma)$ into cycles is unique and we denote by $c(\pi, \sigma)$ the number of the cycles in $G(\pi, \sigma)$.

4.3 Approximation algorithms

We present now approximation algorithms for computing the reversal and transposition distance of two permutations. We will give a 3-approximation algorithm for the unsigned case and a 2-

approximation algorithm for the signed case.

Let us begin with the unsigned case. Note that the only permutation having 0 breakpoints with respect to σ is exactly σ, and then the sequence of reversals and transpositions transforming π into σ must take the number of breakpoints from $b(\pi, \sigma)$ to 0. We also observe that reversals can remove at most two breakpoints, and transpositions can remove at most three breakpoints. This observation implies immediately a lower bound, given in the next theorem.

Theorem 4.3.1 *Given two unsigned permutations π and σ we have*

$$\frac{b(\pi, \sigma)}{3} \leq d(\pi, \sigma).$$

Theorem 4.3.2 *Given two permutations π and σ, with $\pi \neq \sigma$, there is an operation ρ removing at least one breakpoint.*

Proof: The intuitive idea is to increase the first strip on each operation, removing its rightmost breakpoint without introducing new breakpoints.

The first strip on the left is always an increasing strip. Taking the maximum element on this first strip, find its successor, which will be necessarily to the right. If the successor is in the beginning of a strip, or is the only element on the strip, we apply a transposition. If it is in the end, we apply a reversal. ■

Repeated application of Theorem 4.3.2 gives a 3-approximation algorithm for computing the reversal and transposition distance of unsigned permutations. Its time complexity is $O(n^2)$, where n is the size of the permutations. It takes time $O(n)$ to find the operation and apply it.

4.3.1 Signed Permutations

Now we turn to the signed case. Note that the diagram $G(\sigma, \sigma)$ is the only one having $n + 1$ cycles. So, the sequence of reversals and transpositions transforming π into σ must take the number of cycles from $c(\pi, \sigma)$ to $n + 1$. For two permutations π and σ, and an operation ρ, denote $\Delta c(\rho) = c(\rho \cdot \pi, \sigma) - c(\pi, \sigma)$ as the gain in the number of cycles due to an operation ρ.

Lemma 4.3.1 $\Delta c(\rho) \in \{-2, -1, 0, 1, 2\}$

Proof: We note first that ρ can be a reversal or a transposition.

Each reversal acts on two reality edges belonging to at most two cycles, creating or destroying at most one cycle. Hannenhalli and Pevzner [64] have shown that, for a reversal, $\Delta c(\rho) \in \{-1, 0, 1\}$.

Each transposition acts on three reality edges belonging to at most three cycles. Bafna and Pevzner [9] have shown that for the unsigned case $\Delta c(\rho) \in \{-2, 0, 2\}$. It corresponds, in the signed case, to a diagram generated by a permutation composed only by positive strips.

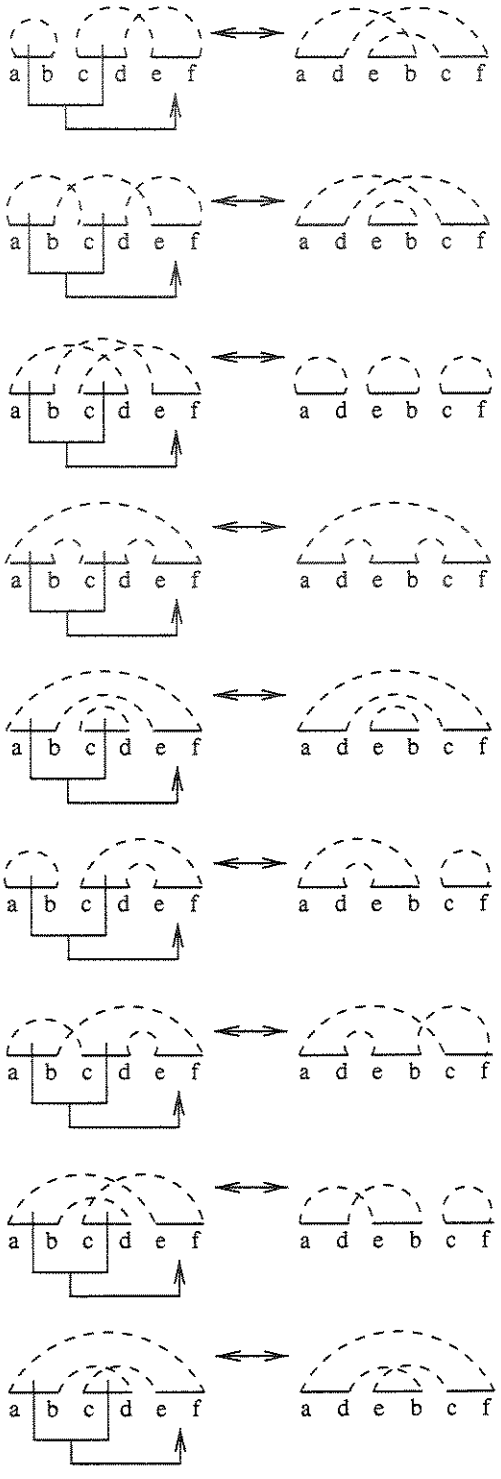

Figure 4.3: This figure shows all possible cases of transposition acting on a signed permutation, where only the affected cycles are shown.

However, in the signed case, we have also $\Delta c(\rho) = -1$ or $+1$. This can be seen from Figure 4.3, which shows all possible actions of a transposition on signed permutations. ■

The following theorem comes directly from Lemma 4.3.1.

Theorem 4.3.3 *Given two signed permutations π and σ then we have*

$$\frac{(n+1) - c(\pi, \sigma)}{2} \leq d(\pi, \sigma)$$

For $x \in \{2, 1, 0, -1, -2\}$, define a x-**move** on π with respect to σ as an operation ρ such that $\Delta c(\rho) = x$. As we mentioned, Figure 4.3 shows all possible actions on signed permutations. In each of the cases shown, a transposition transforms reality edges $(b, a), (d, c)$ and (f, e) into $(d, a), (b, e)$ and (f, c). Dashed lines denote a path that can be formed by one or more desire/reality edges. Since the inverse of a transposition is a transposition, the transformations are reversible. Notice that there is only one pattern corresponding to a 2-move, and only three patterns corresponding to an 1-move. This fact leads to the following theorem.

Theorem 4.3.4 *A diagram admits a 2-move if and only if there are three reality edges (a, b), (c, d), and (e, f) such that*

1. *they appear in this order in the diagram*
2. *they belong to the same cycle*
3. *a is connected to d, b to e, and c to f, by paths not containing any of the edges (a, b), (c, d), and (e, f).*

We show now a way to apply a reversal or a transposition on a signed permutation in order to obtain an increase of $c(\pi, \sigma)$ by at least 2 in two consecutive moves.

Theorem 4.3.5 *Given two signed permutations π and σ, there is either a 1-move, a 2-move or a 0-move followed by a 2-move.*

Proof:

If there are negative strips, Hannenhalli and Pevzner [64] have shown that there is always a reversal increasing the number of cycles, and is therefore an 1-move.

If all strips are positive, we can view this permutation as an unsigned one, and apply a result from Bafna and Pevzner [9], guaranteeing the existence of either a 2-move, or a 0-move followed by a 2-move. ■

From Theorem 4.3.5 we can derive an upper bound for the reversal and transposition distance.

Theorem 4.3.6 *Given two signed permutations π and σ we have*

$$d(\pi, \sigma) \leq (n+1) - c(\pi, \sigma)$$

Given a permutation π to be transformed into σ, the intuitive idea of the algorithm is, while we have negative strips on π with respect to σ we apply reversals as described on Theorem 4.3.5. If we cannot apply reversals of this kind, and this sequence of reversals did not transform π into σ then the diagram is generated by a permutation having only positive strips with respect to σ. Then we use the results of Bafna and Pevzner [9] to discover the sequence of transpositions to be applied. We note that, when these transpositions are being applied, all diagrams are generated from permutations having only positive strips with respect to σ.

This gives a 2-approximation algorithm for computing the reversal and transposition distance of signed permutations. Its time complexity is $O(n^2)$, where n is the size of the permutations. Both a suitable reversal and a suitable transposition, as specified in the proof of Theorem 4.3.5, can be found in time $O(n)$ [9, 64].

4.4 Reversal and transposition diameter

In this section we give initial steps for computing the maximum number of operations for the reversal and transposition distance of signed permutations.

Taking S_n as the set of all permutations with size n, define

$$D(n) = \max_{\pi,\sigma \in S_n} d(\pi, \sigma)$$

to be the **reversal and transposition diameter** of this set. Let $\pi = (-1\ -2\ \ldots\ -(n-1)\ -n)$ and $\sigma = (+1\ +2\ \ldots\ +(n-1)\ +n)$. Then $c(\pi, \sigma) = 1$ for all n, and we have a lower bound (by Theorem 4.3.3),

$$\left\lceil \frac{n}{2} \right\rceil \leq d(\pi, \sigma).$$

More precisely, we can prove the following result.

Theorem 4.4.1 *Taking $\pi = (-1\ -2\ \ldots\ -(n-1)\ -n)$ and $\sigma = (+1\ +2\ \ldots\ n-1\ n)$, then we have*

$$d(\pi, \sigma) = \begin{cases} \left\lfloor \frac{n}{2} \right\rfloor + 1 & \text{if } n = 1, 2 \\ \\ \left\lfloor \frac{n}{2} \right\rfloor + 2 & \text{if } n \geq 3 \end{cases}$$

Proof: We have two cases according to the parity of n.

- When n is odd, a lower bound is $\lceil \frac{n}{2} \rceil \leq d(\pi, \sigma)$. But $\lceil \frac{n}{2} \rceil = \frac{n+1}{2} = \lfloor \frac{n}{2} \rfloor + 1 \leq d(\pi, \sigma)$. To achieve the lower bound, all operations applied must be 2-moves, except one which must be a 1-move. No 2-moves exist in the original diagram, and for every 1-move in the first step, the resulting diagram does not admit 2-moves. Hence, the lower bound cannot be achieved and we have $d(\pi, \sigma) \geq \lfloor \frac{n}{2} \rfloor + 2$.

- When n is even, Tables 4.1 summarize the argument. The table labeled "FIRST MOVE" analyzes all possibilities for the first move. Two of these possibilities require an analysis of the second move as well, which is done in the table labeled "SECOND MOVE." Two arguments are used heavily in this table. One of them is that when negative elements remain, we cannot achieve the $\lfloor \frac{n}{2} \rfloor + 1$ lower bound because we are forced to use at least one reversal, which is never a 2-move. The other is that one can verify in some cases that a 2-move does not exist looking for the characterization given by Theorem 4.3.4.

When $n \geq 3$ we can obtain an upper bound for $d(\pi, \sigma)$ in the following way. First, we apply a reversal on π, obtaining $r(1, n) \cdot \pi = (+n \ +(n-1) \ \ldots \ +2 \ +1)$. Then we use the result of Christie [30], also obtained independently by Meidanis, Walter and Dias [93], that determines the transposition distance $d_t(r \cdot \pi, \sigma) = \lfloor \frac{n}{2} \rfloor + 1$, for $n > 2$. Therefore, we have the upper bound, $d(\pi, \sigma) \leq \lfloor \frac{n}{2} \rfloor + 2$, for $n \geq 3$. This completes the proof. ■

We can verify that $D(n) = \lfloor \frac{n}{2} \rfloor + 2$ for $n = 3, 4$.

4.5 Conclusions

In this article we have presented approximation algorithms for computing the reversal and transposition distance. For the signed and unsigned cases we have shown algorithms based on the notion of breakpoints and cycles, respectively. For the signed case our algorithm uses a specific type of reversal while possible, and after that it uses part of the Bafna and Pevzner theory [9] to get the transpositions to be applied.

The lower bounds used to estimate the approximation factor were simple, yet they lead to a deeper result, namely, the calculation of the exact distance between permutation $(-1 \ -2 \ \ldots \ -(n-1) \ -n)$ and the identity. This proof is more involved and uses the characterization of 2- and 1-moves given in Figure 4.3. Of course the result provides a lower bound for the diameter, which we conjecture to be an upper bound as well.

Plans for future work include dealing with other operations, notably the combined reversal+transposition, which is a natural operation to consider form the biological standpoint, and studying weighted problems, where each type of operation has a different weight, and the goal is to minimize the total weight.

FIRST MOVE		
reversal	2-move	impossible, no reversal is a 2-move
	1-move	must be $r(i,j)$ with $j-i$ even; analyze second move
	0-move	if $r(1,n)$, use known result on transposition distance (a); otherwise negative elements remain
transposition	2-move	the unique 2-move pattern in Figure 4.3 does not exist in the diagram
	1-move	must be $t(i,j,k)$ with $j-i$ and $k-i$ both odd; analyze second move
	0-move	negative elements remain

SECOND MOVE		
$r(i,j)$ with $j-i$ even	2-move	the unique 2-move pattern in Figure 4.3 does not exist in the diagram
	1-move	if transposition, negative elements remain; if reversal, negative elements remain except when first move was $r(1,n)$ or $r(i+1,n)$ for odd i, but then the unique 2-move pattern in Figure 4.3 does not exist in the diagram
$t(i,j,k)$ with both $j-i$ and $k-j$ odd	2-move	the unique 2-move pattern in Figure 4.3 does not exist in the diagram
	1-move	if transposition, negative elements remain; if reversal, negative elements remain except when it is $r(1,n)$. But then $r(1,n) \cdot t(i,j,k) = t(n+2-k, n+2-j, n+2-i) \cdot r(1,n)$, which was already analyzed

Table 4.1: Analysis of the first two steps in computing the distance. (a) See result by Christie and Meidanis, Walter, Dias in the text. In the SECOND MOVE table we do not consider 0-moves since if the second move is a 0-move, the lower bound cannot be achieved, because the first move was an 1-move.

Capítulo 5

A Lower Bound on the Reversal and Transposition Diameter *

João Meidanis

Maria Emilia M. T. Walter

Zanoni Dias

Abstract

One possible model to study genome evolution is to represent genomes as permutations of genes and compute distances based on the minimum number of certain operations (rearrangements) needed to transform one permutation into another. Under this model, the shorter the distance, the closer the genomes are. Two operations that have been extensively studied are the reversal and the transposition. A reversal is an operation that reverses the order of the genes on a certain portion of the permutation. A transposition is an operation that "cuts" a certain portion of the permutation and "pastes" it elsewhere in the same permutation. In this paper we show that the reversal and transposition distance of the signed permutation $\pi_n = (-1\ -2\ \ldots\ -(n-1)\ -n)$ with respect to the identity is $\lfloor n/2 \rfloor + 2$ for all $n \geq 3$. We conjecture that this value is the diameter of the permutation group under these operations.

* *Trabalho depositado como Relatório Técnico no Instituto de Computação da Unicamp em outubro de 2000, sob o número IC-00-16. Um resumo deste relatório foi publicado no volume 9, ano 5, de novembro de 2002, do* **Journal of Computational Biology.**

5.1 Introduction

One possible model to study genome evolution is to represent genomes as **permutations** of genes and compute distances based on the minimum number of certain **operations** (rearrangements) needed to transform one permutation into another. Under this model, the shorter the distance, the closer the genomes are.

In general, genes are represented as integers from 1 to n, and a permutation $\pi : \{1, 2, \ldots, n\} \mapsto \{1, 2, \ldots, n\}$ by

$$(\pi_1 \ \pi_2 \ \ldots \ \pi_n),$$

where π_i denotes $\pi(i)$.

Permutations can be **signed**, in which case each π_i has a positive or negative sign to model the orientation of genes. We will call **permutation group** the set of all permutations of a given size n. The unsigned permutation group has $n!$ elements, while the signed group has $2^n n!$ elements.

In this note we are interested in the diameter of permutation groups, that is, the maximum distance possible between two permutations of size n, under several operation choices. Two operations that have been extensively studied are the *reversal* and the *transposition*. A **reversal** is an operation that reverses the order of the genes on a certain portion of the permutation. A **transposition** is an operation that "cuts" a certain portion of the permutation and "pastes" it elsewhere in the same permutation. (Refer to Section 5.2 for more formal definitions.) A transposition is also called a **block move** in the literature. A **block interchange** operation exchanges two portions of a permutation [28]. Transpositions and block interchanges never affect the signs (if present) of a permutation. For this reason, they are studied in the unsigned case only. We could also conceive an operation that "cuts" a portion and "pastes" it elsewhere reversed. Call this a **transversal**.

Table 5.1 shows what is currently known about the diameter for signed and unsigned permutations under various combinations of the above operations. In this note we provide a lower bound for the diameter in the case of signed permutations evolving by transpositions and reversals.

Analyzing genomes evolving due to different mutational events represents today a great challenge. Hannenhalli and others [61] analyzed genomes evolving by different events, particularly reversals and transpositions. Hannenhalli and Pevzner [62] presented a polynomial time algorithm for comparing genomes evolving by reversals, translocations, fusions and fissions. Gu, Peng and Sudborough [56, 55] gave approximation algorithms to compute the distance between two signed permutations, allowing three operations, reversal, transposition and transversal.

In this work we contribute to the analysis of reversals and transpositions acting on a single chromosome having genes with known orientation. We show a permutation π_n that needs at least $\lfloor n/2 \rfloor + 2$ steps to be sorted, thus obtaining a lower bound on the diameter of the signed

Operations	Size	Degree	Diameter
Reversals (unsigned) [8]	$n!$	$\binom{n}{2}$	$D = n - 1$
Reversals (signed) [64, 95]	$2^n n!$	$\binom{n}{2} + n$	$D = n + 1$
Transpositions [9, 93, 30]	$n!$	$\binom{n+1}{3}$	$\lfloor n/2 \rfloor + 1 \leq D \leq \lfloor 3n/4 \rfloor$
Reversals, transpositions *	$2^n n!$	$\binom{n+1}{3} + \binom{n}{2} + n$	$\lfloor n/2 \rfloor + 2 \leq D$
Block interchange [28]	$n!$	$\binom{n+1}{4}$	$D = \lfloor n/2 \rfloor$

* This paper

Table 5.1: Results known about the diameter of permutation groups under genome rearrangement operations. The column "Size" refers to the size of the graph, i.e., the total number of permutations for n elements. "Degree" is how many neighbors a permutation has. In the column "Diameter" either the diameter is given or the known bounds, with D representing the diameter.

permutation group under these operations.

5.2 Definitions

In this section we formalize the problem of computing the reversal and transposition distance of linear chromosomes.

We assume that the order and orientation of genes in a chromosome are represented by a permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, where each π_i is a signed integer such that $1 \leq |\pi_i| \leq n$ and $|\pi_i| \neq |\pi_j|$ for $i \neq j$.

A reversal $r(i, j)$ is defined by two integers i, j, such that $1 \leq i \leq j \leq n$, reversing the order and sign of π_k, $i \leq k \leq j$. Thus we have

$$r(i,j) \cdot (\pi_1\ \ldots\ \pi_{i-1}\ \pi_i\ \ldots\ \pi_j\ \pi_{j+1}\ \ldots\ \pi_n) =$$

$$(\pi_1\ \ldots\ \pi_{i-1}\ \overline{\pi}_j\ \ldots\ \overline{\pi}_{i+1}\ \overline{\pi}_i\ \pi_{j+1}\ \ldots\ \pi_n)$$

where $\overline{\pi}_k$ means π_k with opposite sign.

A transposition $t(i, j, k)$ is defined by three integers i, j, and k such that $1 \leq i < j \leq n+1$, and $k \notin [i, j]$, in the following way. It "cuts" the portion between positions i and $j-1$, including the extremes, and "pastes" it just before position k. Thus, if $i < j < k$, we can write

$$t(i,j,k) \cdot (\pi_1\ \ldots\ \pi_{i-1}\ \pi_i\ \ldots\ \pi_{j-1}\ \pi_j\ \ldots\ \pi_{k-1}\ \pi_k\ \ldots\ \pi_n) =$$

$$(\pi_1\ \ldots\ \pi_{i-1}\ \pi_j\ \ldots\ \pi_{k-1}\ \pi_i\ \ldots\ \pi_{j-1}\ \pi_k\ \ldots\ \pi_n)$$

Given two permutations π and σ, we want to compute a shortest series of reversals and transpositions that transforms π into σ, that is, we want to find $\varrho_1, \varrho_2, \ldots, \varrho_u$, where ϱ_i is a

reversal or a transposition, such that $\varrho_u \cdot \varrho_{u-1} \cdot \ldots \cdot \varrho_2 \cdot \varrho_1 \cdot \pi = \sigma$ and u is minimum. We call u the **reversal and transposition distance** between π and σ and denote it by $d(\pi, \sigma)$. Without loss of generality we can fix σ. All our developments will be done with σ being the identity permutation, which is $\sigma = \iota_n = (+1 \ \ldots \ +n)$. In this case we denote $d(\pi, \iota_n)$ simply by $d(\pi)$.

In the following an **operation** can be a reversal or a transposition.

A powerful tool for studying the reversal and transposition distance is the *reality and desire diagram* of two permutations. In the literature [9, 64, 63] this is called the *breakpoint graph* of two permutations, but we prefer to call it a diagram because its graph structure alone does not capture all the important information: the order of nodes is relevant too.

We first extend a permutation π by adding $\pi_0 = +0$ and $\pi_{n+1} = +(n+1)$. The extended permutation will still be denoted by π. We construct this diagram writing the original permutation π in the following way. Replace each integer i by a pair of points $-i$ and $+i$, in this order. For instance $+4$ is replaced by -4 and $+4$; -7 is replaced by $+7$ and -7. Add two extra points, one called $+0$ at the beginning of the sequence, and one called $-(n+1)$ at the end of the sequence. Now draw **reality** edges between $+0$ and $-\pi_1$, between $+\pi_{i-1}$ and $-\pi_i$, and between $+\pi_n$ and $-(n+1)$. Finally, draw **desire** edges between $+0$ and -1, between $+(i-1)$ and $-i$, and between $+n$ and $-(n+1)$ (see Figure 5.1). Again, in the literature, reality edges are called *black edges* and desire edges are called *gray edges*. We prefer the denominations reality and desire because they are more informative: reality edges refer to the current permutation, that is, where we are, and desire edges refer to the target permutation, that is, where we would like to be. We denote $G(\pi)$ the diagram of the permutation π (with respect to the identity).

Observe that the diagram is composed of a number of cycles, with each cycle alternating between reality and desire edges. The **length** of a cycle is the number of reality edges in it (which is the same as the number of desire edges in it). The decomposition of $G(\pi)$ into cycles is unique and we denote by $c(\pi)$ the number of cycles in $G(\pi)$.

The cycles of $G(\pi)$ are denoted by a bracket notation as follows. We number the reality edges of $G(\pi_n)$ from 1 to $n+1$ by assigning label i to the reality edge (π_i, π_{i-1}), with $1 \leq i \leq n+1$. Besides, we will assign to the label i from cycle c an orientation $+i$ or $-i$, defined with respect to the orientation of the greatest (in absolute value) label l from c, which is $+l$ by convention. So, taking these labels and their orientations, in the order they appear in around the cycle, the unique cycle of the diagram in Figure 5.1 (c) is

$$[+(n+1), +(n-1), \ldots, +4, +2, -1, -3, \ldots, -n]$$

for n odd or

$$[+(n+1), +(n-1), \ldots, +3, +1, -2, -4, \ldots, -n]$$

for n even.

The diagram has exactly $n+1$ reality edges and the same number of desire edges. The idea is that reality edges indicate the situation as it is now, and desire edges indicate the situation

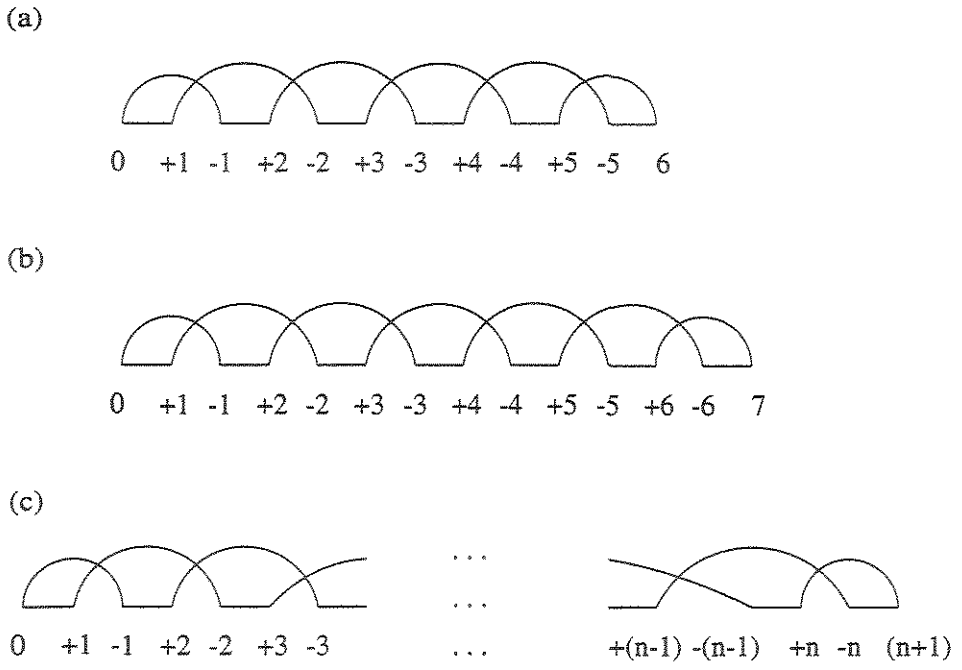

Figure 5.1: The figure shows: (a) The diagram for $\pi = (-1\ -2\ -3\ -4\ -5)$. (b) The diagram for $\pi = (-1\ -2\ -3\ -4\ -5\ -6)$. (c) The general diagram for $\pi = (-1\ -2\ \ldots\ -(n-1)\ -n)$, for all n.

sought. When reality equals desire in all edges, we have $\pi = \iota_n$ and $d(\pi) = 0$. Therefore, our goal is to apply reversals and transpositions so that reality becomes desire.

Note that the diagram $G(\iota_n)$ is the only one having $n+1$ cycles. So, the sequence of reversals and transpositions transforming π into ι_n must take the number of cycles from $c(\pi)$ to $n+1$. For a permutation π, and an operation ϱ, denote by $\Delta c(\pi, \varrho)$ the difference $c(\varrho \cdot \pi) - c(\pi)$. This is the gain in the number of cycles due to operation ϱ applied to π.

Theorem 5.2.1 $\Delta c(\pi, \varrho) \in \{-2, -1, 0, 1, 2\}$

Proof: We note first that ϱ can be a reversal or a transposition.

Each reversal acts on two reality edges belonging to at most two cycles, creating or destroying at most one cycle. Hannenhalli and Pevzner [64] have shown that, for a reversal, $\Delta c(\pi, \varrho) \in \{-1, 0, 1\}$.

Each transposition acts on three reality edges belonging to at most three cycles. Figure 5.2 shows all possible actions of a transposition on a signed permutation. As we can see, there are cases where the number of cycles stays the same, or increases by one or two. So, for a transposition, $\Delta c(\pi, \varrho) \in \{-2, -1, 0, 1, +2\}$. ■

For $x \in \{-2, -1, 0, 1, 2\}$, define an x-move on π as an operation ϱ such that $\Delta c(\pi, \varrho) = x$. Notice that, in Figure 5.2, there is only one pattern corresponding to a 2-move (or -2-move),

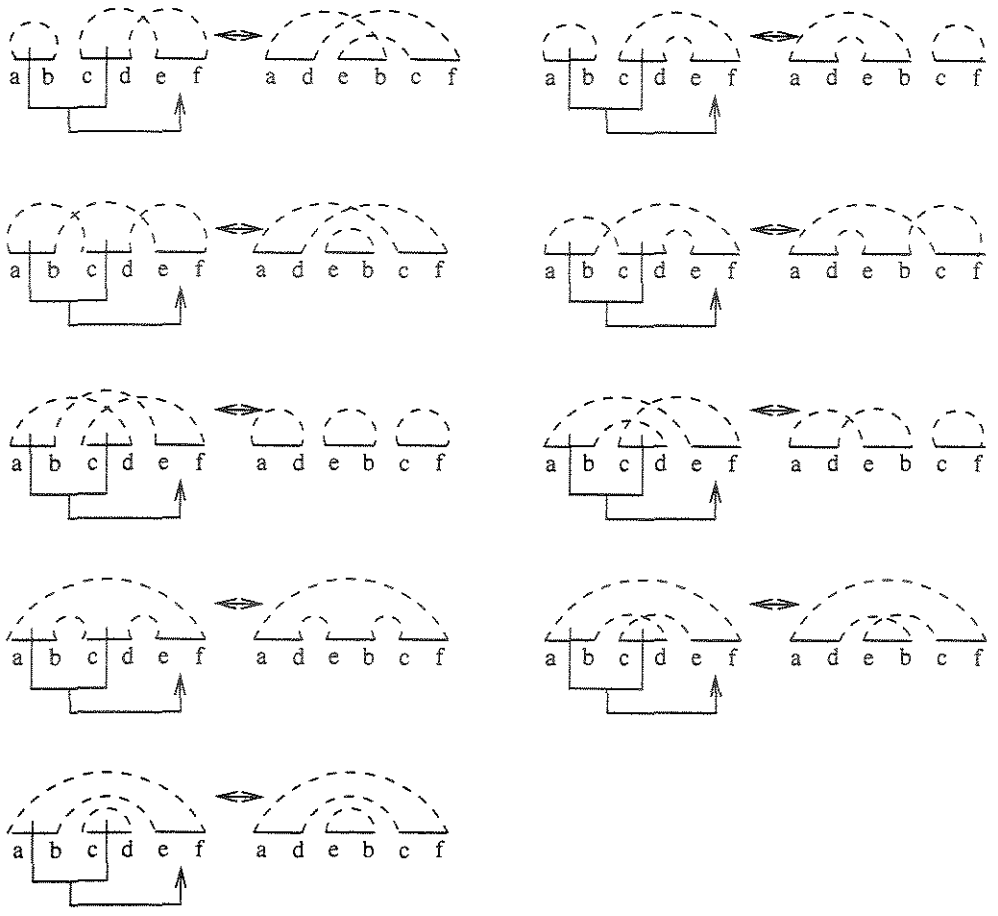

Figure 5.2: This figure shows all possible actions of a transposition on a signed permutation. Only affected cycles are shown. A dashed line indicates a path formed by one or more desire and reality edges. Since the inverse of a transposition is a transposition, the transformations are reversible.

three patterns corresponding to a 1-move (or -1-move), and the others corresponding to 0-moves.

The entire distance problem can be seen as finding shortest paths in a directed graph where a vertex corresponds to a permutation π, and there is an edge (π, σ) from π to σ when there is an operation (reversal or transposition) ϱ such that $\sigma = \varrho \cdot \pi$. We are interested in shortest directed paths from π to ι_n, where the length of a path is just its number of edges. However, we can assign weights to the edges in a way that will help us investigate the problem. In the sequel we will define the weight of an edge and of a path, on that graph.

Definition 5.2.1 *Given the permutations π and σ, such that $\sigma = \varrho \cdot \pi$ for some operation ϱ, we define the* **weight** *of the edge (π, σ) as*

$$w(\pi, \sigma) = 2 + c(\pi) - c(\sigma).$$

Notice that $w(\pi, \sigma) \geq 0$ for all edges (Theorem 5.2.1). The weight $w(\pi, \sigma)$ can be also written as $2 - \Delta c(\pi, \varrho)$, where ϱ is the operation that transforms π into σ. Since 2 is the highest value that $\Delta c(\pi, \varrho)$ can take, and we know that high values of $\Delta c(\pi, \varrho)$ will get us closer to our goal, we can think of the weight as a measure of "waste" in each operation we do.

Definition 5.2.2 *Given a path $p = \pi_0\pi_1\pi_2 \ldots \pi_{k-1}\pi_k$, we define the* **weight** *of p as*

$$w(p) = \sum_{i=1}^{k} w(\pi_{i-1}, \pi_i).$$

Note that $w(p) \geq 0$ for all paths. We can now relate the length of a path with the weight of the same path, with important consequences on the distance. Let $|p|$ denote the length of a path p.

Theorem 5.2.2 *Let $p = \pi_0\pi_1\pi_2 \ldots \pi_{k-1}\pi_k$ be a path. We have*

$$w(p) = 2|p| + c(\pi_0) - c(\pi_k).$$

The proof is just an induction on k. An important corollary is the following.

Corollary 5.2.1 *For any permutation π and any shortest path p from π to ι_n we have*

$$d(\pi) = \frac{w(p) - c(\pi) + n + 1}{2}.$$

The proof is immediate from the theorem, using $\pi_0 = \pi$ and $\pi_k = \iota_n$.

5.3 The reversal and transposition diameter

Taking S_n as the set of all signed permutations with size n, define $D(n) = \max_{\pi \in S_n} d(\pi, \iota_n)$ as the **reversal and transposition diameter** of signed permutations. In this section we present a lower bound on this number, based on the distances of particular permutations for each integer n.

This particular permutation is $\pi_n = (-1\ -2\ \ldots\ -(n-1)\ -n)$. We will compute its reversal and transpositions distance, which will give a lower bound for the diameter $D(n)$. We start by showing an upper bound for $d(\pi_n)$, for all $n \geq 3$.

Theorem 5.3.1 *We have $d(\pi_n) \leq \left\lfloor \frac{n}{2} \right\rfloor + 2$ for $n \geq 3$.*

Proof: First we apply the reversal $r(1,n)$ on π_n, obtaining

$$\pi'_n = r(1,n) \cdot \pi_n = (+n\ +(n-1)\ \ldots\ +2\ +1),$$

a permutation with positive signs only.

After that we recall a result from Meidanis, Walter and Dias [93], independently shown by Christie [30], proving that the transposition distance $d_t(\pi'_n)$ is $\lfloor \frac{n}{2} \rfloor + 1$, for $n > 2$.

The total number of operations is then $\lfloor \frac{n}{2} \rfloor + 2$, which is an upper bound on the distance $d(\pi_n)$ for $n \geq 3$. ■

Our strategy to show that this upper bound is a lower bound as well will be to prove that every path p from π_n to ι_n satisfies $w(p) \geq 3$. Then, by force of Corollary 5.2.1 and Lemma 5.3.1, we will have the desired result (see Theorem 5.3.4).

The general form of the diagram generated by this permutation is given in Figure 5.1 (c). The number of cycles is always 1, and we state this as our next lemma.

Lemma 5.3.1 *We have $c(\pi_n) = 1$ for all n.*

We need auxiliary results to support our claims. One that appears with frequency is a sufficient condition for the lack of 2-moves. Recall the format of the cycles in the diagram of π_n:

$$c = [+(n+1), +(n-1), \ldots, +2, -1, -3, \ldots, -n],$$

for n odd and

$$c = [+(n+1), +(n-1), \ldots, +3, +1, -2, \ldots, -n]$$

for n even. Notice that regardless of the parity of n these cycles are formed by two decreasing subsequences. We call **bimonotonous** the cycles formed by two decreasing subsequences, the first made of positive elements, and the second formed by negative elements. Such cycles cannot be broken by a transposition, as the following results show.

Lemma 5.3.2 *A permutation π admits a 2-move if and only if there are three reality edges labeled i, j and k with $i < j < k$, belonging to the same cycle in $G(\pi)$, and appearing in that cycle either in the order k, i, j (or i, j, k or j, k, i) with orientation $+$, or in the order k, j, i (or j, i, k or i, k, j) with orientation $-$.*

Proof: Theorem 5.2.1 shows that there is just one pattern corresponding to a 2-move. In this pattern (see Figure 5.2), we can verify that, taking the three labels (belonging to the same cycle) i, j and k such that $i < j < k$, and assigning to label k the orientation $+k$, we force the orientations of i and j to be respectively $+i$ and $+j$, implying that these three labels appear in the cycle with the order k, i, j (or i, j, k or j, k, i), and all three with the same orientation. Analogously, if we assign to k the orientation $-k$, the orientations of i and j become $-i$ and $-j$, implying the order k, j, i (or j, i, k or i, k, j), with i, j and k with the same orientation.

The proof on the other side is immediate. We apply $t(i,j,k)$ on π, with i, j and k following the conditions of the lemma, and obtain the desired result. ■

Theorem 5.3.2 *Let π be a permutation for which all cycles in its reality and desire diagram are bimonotonous. Then $w(\pi, \varrho\pi) \geq 1$ for all operations ϱ.*

Proof: Of course, $w(\pi, \varrho\pi) = 0$ is equivalent to saying that ϱ is a 2-move on π. A 2-move has to be a transposition, and acting on three reality edges of the same cycle. However, by the bimonotonicity of the cycles of π, we cannot choose three labels following the conditions of Lemma 5.3.2, considering just one of these two subsequences. Another way to get these labels would be to choose them from both subsequences. But then they will not have the same orientation, so also in this case we cannot have the conditions of Lemma 5.3.2. ■

We are now ready for our main theorem.

Theorem 5.3.3 *Let $p = \sigma_0\sigma_1 \ldots \sigma_k$ be any path from $\pi_n = \sigma_0$ to $\iota_n = \sigma_k$. Then we have, for $n \geq 3$:*

1. $w(\sigma_0\sigma_1) \geq 1$
2. *if* $w(\sigma_0\sigma_1) = 1$, *then* $w(\sigma_1\sigma_2) \geq 1$
3. *if* $w(\sigma_0\sigma_1\sigma_2) = 2$, *then* $w(\sigma_2 \ldots \sigma_k) \geq 1$

Proof: The first claim is true because the weight is always greater than or equal to zero, and it is zero only if the operation is a 2-move. However, $\sigma_0 = \pi_n$ has only one cycle, and this cycle is bimonotonous. Our claim then follows from Theorem 5.3.2.

For the second claim, observe that $w(\sigma_0\sigma_1) = 1$ exactly when the operation ϱ that acted on $\sigma_0 = \pi_n$ was a 1-move. Both reversals and transpositions can be 1-moves in the signed case, so we need to analyze these two cases.

Let us deal with reversals first. It is well known [106] that a reversal breaks a cycle (that is, is a 1-move) if and only if the two reality edges where it acts have opposite orientations. Since $r(i, j)$ acts on reality edges i and $j + 1$, this means that $r(i, j)$ is a 1-move if and only if i and j have the same parity. The diagram of the permutation $r(i, j) \cdot \pi_n$ in this case has two cycles. The exact pattern of the resulting cycles depends on the relative parity of i, j, and n, but in all cases they are bimonotonous. For instance, if i, j, and n are all odd, these cycles are

$$\begin{aligned} c_1 &= [+j, +(j-2), \ldots, +(i+2), +i, +(i-2), \ldots, +3, +1, \\ &\quad -2, -4, \ldots, -(i-1)], \\ c_2 &= [+(n+1), +(n-1), \ldots, +(j+1), +(j-1), +(j-3), \ldots, +(i+1), \\ &\quad -(j+2), -(j+4), \ldots, -n]. \end{aligned}$$

It is apparent that these two cycles are bimonotonous. The other cases can be verified analogously. Therefore $w(\sigma_1\sigma_2) \geq 1$ if ϱ is a reversal.

The case where ϱ is a transposition $t(i,j,k)$ also requires an analysis based on the parity of i, j, k, and n. From Figure 5.1 and Figure 5.2 we see that this operation is a 1-move if and only if i and k have the same parity, and j has the opposite parity from i and k. For instance, in the case of i, k, and n all odd and j even, we have a diagram $G(t(i,j,k)\cdot\pi_n)$ formed by two cycles, which are

$$\begin{aligned} c_1 &= [+(k-1), +(k-3), \ldots, +j, +(j-2), \ldots, +(i+1)] \\ c_2 &= [+(n+1), +(n-1), \ldots, +(k+1), +(i+k-j-1), \\ &\quad +(i+k-j-3), \ldots, +(i+2), +i, +(i-2), \ldots, +3, +1, \\ &\quad -2, -4, \ldots, -(i-1), -(i+k-j+1), \\ &\quad -(i+k-j+3), \ldots, -k, -(k+2), \ldots, -n]. \end{aligned}$$

The first cycle is monotonous and therefore does not admit a 2-move. The second cycle is bimonotonous, and, by Theorem 5.3.2, does not admit a 2-move either. The other cases can be verified analogously.

Let us now turn to the third claim. Again we divide the proof into two cases: either there is a negative element in σ_2 or all elements there are positive. If there is at least one negative element, then $w(\sigma_2 \ldots \sigma_k) \geq 1$ because otherwise only transpositions would be applied until we reach ι_n, but ι_n does not have negative elements and transpositions do not change signs.

We concentrate then in the case where σ_2 has all elements positive. Since $\sigma_0 = \pi_n$ has all elements negative, there are only four possible ways of reaching an all-positive permutation in two steps:

1. $\sigma_2 = r(1,i)\cdot r(i+1,n)\cdot\sigma_0$, for some i between 1 and $n-1$, including extremes.
2. $\sigma_2 = r(i+1,n)\cdot r(1,i)\cdot\sigma_0$, for some i between 1 and $n-1$, including extremes.
3. $\sigma_2 = t(i,j,k)\cdot r(1,n)\cdot\sigma_0$, for some triple i, j, k with $1 \leq i < j < k \leq n+1$.
4. $\sigma_2 = r(1,n)\cdot t(i,j,k)\cdot\sigma_0$, for some triple i, j, k with $1 \leq i < j < k \leq n+1$.

The first two cases are actually the same, since $r(1,i)$ and $r(i+1,n)$ commute. In fact, we will show that all cases can be reduced to the third one. The key to this fact is to notice that any transposition can be written as the product of three reversals:

$$t(i,j,k) = r(i,k-1)\cdot r(i,j-1)\cdot r(j,k-1). \tag{5.1}$$

This can be easily verified from the definitions. If we use $i=1$ and $k=n+1$ in this equation, we get:

$$t(1,j,n+1) = r(1,n)\cdot r(1,j-1)\cdot r(j,n),$$

showing that Cases 1 and 2 are indeed particular instances of Case 4 (recall that $r(1,n)^2 = I$). On the other hand,

$$r(1,n) \cdot t(i,j,k) \cdot r(1,n) = t(k',j',i'),$$

where $i' = n+2-i$, $j' = n+2-j$, and $k' = n+2-k$, which shows that Case 4 can be reduced to Case 3.

Let us then concentrate on Case 3. Notice that in this case σ_1 is the permutation $(+n\ +(n-1)\ \ldots\ +2\ +1)$. A consequence of the work by Meidanis, Walter, and Dias [93] and that of Christie [30], which computed the transposition distance of such permutations, is that $w(\sigma_1 \ldots \iota_n) \geq 2$ for any path consisting of transpositions only. Now if $w(\sigma_2 \ldots \sigma_k) = 0$, this would refer to a path using transpositions only, and therefore we can conclude that $w(\sigma_1\sigma_2) = 2$ and that $w(\sigma_0\sigma_1) = 0$, a contradiction since the first step $r(1,n)$ was a reversal. It follows that $w(\sigma_2 \ldots \sigma_k) \geq 1$ as claimed. ■

Theorem 5.3.4 *We have* $d(\pi_n) \geq \lfloor \frac{n}{2} \rfloor + 2$ *for* $n \geq 3$.

Proof: Theorem 5.3.3 guarantees that $w(p) \geq 3$ for any path from π_n to ι_n. Plugging this into the formula of Corollary 5.2.1 we conclude that

$$d(\pi_n) \geq \frac{n+3}{2},$$

which implies $d(\pi_n) \geq \lfloor \frac{n}{2} \rfloor + 2$ since $d(\pi_n)$ is an integer. ■

The next theorem comes directly from Theorems 5.3.4 and 5.3.1.

Theorem 5.3.5 *Given the permutations* π_n *and* ι_n*, for all* n*, then we have*

$$d(\pi_n) = \begin{cases} \lfloor \frac{n}{2} \rfloor + 1 & \text{if } n = 1,2 \\ \lfloor \frac{n}{2} \rfloor + 2 & \text{if } n \geq 3 \end{cases}$$

Proof: For $n = 1$ it is obvious that $d(\pi_n) = 1$ since $\pi_n \neq \iota_n$ and a reversal will do. For $n = 2$ a minimum series of operations transforming π_n into ι_n consists of two operations. For $n \geq 3$, the result follows from Theorems 5.3.1 and 5.3.4. ■

5.4 Conclusions

In this work we extend the analysis of transpositions done by Bafna and Pevzner (1995) to signed permutations, and compute the the reversal and transposition distance of the signed permutation $(-1\ -2\ \ldots\ -(n-1)\ -n)$ with respect to the identity $(+1\ +2\ \ldots\ +n-1\ +n)$. The proof is based on the number of cycles that can be created, on the first two steps, in the diagrams generated on any sequence of operations transforming π_n on ι_n. Obviously this result

gives a lower bound for the diameter. We conjecture that this is also an upper bound. We remark that the exact value of the transposition diameter is still unknown (see Table 5.1).

An interesting point to be studied later is the diameter of signed permutations under reversals, transpositions, and *transversals*. A transversal acts by moving a block of genes to another place on the permutation, but with the genes reversed. This operation is biologically as natural as the transposition.

Another line of study is to consider different weights for transpositions and reversals. With equal weights, as done here, the minimum path consists predominantly of transpositions. It would be interesting to use weights suggested by what has been observed in practice. Apparently, transpositions should weigh about twice as much as reversals.

Parte II

Contribuições Originais

Capítulo 6

An Alternative Algebraic Formalism For Genome Rearrangements *

João Meidanis

Zanoni Dias

Abstract

Here we relate the recent theory of genome rearrangements to the theory of permutation groups in a new way and hope to set the ground for further advances in the area. This work was motivated by the fact that many arguments in genome rearrangements are of the form "look at the figure", and lack more formal algebraic derivation. We intend to give the area a strong algebraic formalism, much as analytic geometry provided an alternative for geometric arguments based on pictures.

*Este trabalho corresponde a versão estendida do artigo apresentado no **Gene Order Dynamics, Comparative Maps and Multigene Families** (DCAF'2000) realizado na cidade de Le Chantecler, no Canadá, em setembro de 2000, e que também integra o livro **Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment and Evolution of Gene Families** publicado em novembro de 2000.*

6.1 Introduction

In this paper we are concerned with the genome rearrangement problem viewed as a combinatorial problem. In the general formulation of this problem we are given two genomes (or parts of genomes), viewed as ordered lists of genes (or others markers), and a set of allowed mutation events (reversals, transpositions, etc). To solve the problem we must find the minimum number of events that lead from one genome to another. In general the solution is symmetric, that is, the same series of events, taken backward, will transform the second genome into the first. We will also restrict ourselves to the case of conservative events, that is, events that do not change the available gene pool. Thus events such as duplications or deletions will not be considered in this study.

Recent developments in this field include the polynomial solution to the signed reversal case [64], the NP-hardness of unsigned reversal distance [21], and partial results for transposition distance [9, 92], to name just a few. Many doctoral dissertations were devoted to this theme (see, for instance, the theses of [117], [30], and [119]). Transposition distance seems to be a harder problem, that has eluded researchers for many years now. Its computational complexity is still unknown. We feel that new, more powerful formal tools are needed to successfully attack this problem.

The mathematical formalization of genome rearrangements usually begins by representing genomes as *permutations*. Thus, a genome π consisting of genes π_1, π_2, π_3, ..., π_n in this order is written as:

$$\pi = (\pi_1\ \pi_2\ \pi_3\ \ldots\ \pi_n) \tag{6.1}$$

meaning that π is the function (permutations are functions):

$$[1 \Rightarrow \pi_1, 2 \Rightarrow \pi_2, 3 \Rightarrow \pi_3, \ldots, n \Rightarrow \pi_n]$$

that is, π maps 1 into π_1, 2 into π_2, and so on.

In this paper we will propose a different view of a genome as a permutation, namely, that Equation [6.1] denotes the function:

$$[\pi_1 \Rightarrow \pi_2, \pi_2 \Rightarrow \pi_3, \ldots, \pi_{n-1} \Rightarrow \pi_n, \pi_n \Rightarrow \pi_1] \tag{6.2}$$

that is, π maps π_1 into π_2, π_2 into π_3, and so on. Note that the last gene π_n is mapped into the first gene π_1. This is necessary, because permutations are functions that map each element into some other, and they cannot repeat images. However, this implies a circular character to our genome. But circular genomes do exist, and, as we will see in subsequent sections, the study of rearrangements of linear genomes is really not much different from circular ones.

Our goal in this note is to convince the reader that interpretation (6.2) is much more sensible, for a number of reasons. First, it allows us to directly apply many long known results from permutation group theory. Important tools such as breakpoints, the breakpoint graph, cycles, good

cycles, bad cycles, gray edges, black edges, which served as basic building blocks for most of the advances in the field can be algebraically defined instead of graphically defined as they have been until now. Therefore, arguments that relied on pictures can now be expressed completely in algebraic terms. We consider this a powerful step towards a massive attack on such problems, much like analytic geometry is a powerful way of looking into geometric problems.

In Section 6.2 we briefly review the basics on permutation groups. Section 6.3 contains the first steps in redefining genome rearrangements under the new formalism that we propose. In the Section 6.4 we use theory just developed to show some results that have been proved based on pictures. In Section 6.5 we apply the theory to reversal distance problem. Finally, we conclude in Section 6.6.

6.2 Permutation Groups

Permutations groups have been studied at least since the eighteenth century, when Galois wrote his much acclaimed theory for solving algebraic equations. Here we briefly recall a few classical results that are useful in genome rearrangements. For more information see references [70, 87].

Given a base set E, a *permutation* on E is a one-to-one function from E onto itself. Permutations are composed of one or more *cycles*. A cycle involving elements a, b, c, for instance, is written:

$$(a\ b\ c)$$

meaning that a is mapped into b, which is mapped into c, which in turn is mapped back into a. Cycles can be of any length. Cycles of length 1 are not explicitly written. Thus, if we write:

$$\alpha = (a\ b\ c)$$

we implicitly mean that all others elements are left in place by α, that is, $\alpha(x) = x$ for $x \neq a, b, c$. Note: $(a\ b\ c) = (b\ c\ a) = (c\ a\ b)$.

The *support* of a permutation α, $Supp(\alpha)$, is the set of elements not fixed by α:

$$Supp(\alpha) = \{x \in E_n | \alpha(x) \neq x\}.$$

The *size* of a cycle is the number of elements in its support.

The *product* (or *composition*) of two permutations α, β is denoted by $\alpha\beta$. In general $\alpha\beta \neq \beta\alpha$, but when α and β are disjoint cycles they commute: $\alpha\beta = \beta\alpha$. Every permutation can be written in an unique way as a product of disjoint cycles (apart from the order of the factors). We refer to this as the *cycle decomposition* of a permutation.

The identity permutation, that maps every element into itself, will be denoted by 1. Every permutation α has an inverse α^{-1} such that $\alpha\alpha^{-1} = \alpha^{-1}\alpha = 1$. For cycles, the inverse is

obtained reverting the order of the elements: $(a\ b\ c)$ is the inverse of $(c\ b\ a)$. For a general permutation, invert every cycle in its cycle decomposition.

To compute the product of α and β, $\alpha\beta$, we must keep in mind that β will be applied first, and then α, as in $\alpha\beta(x) = \alpha(\beta(x))$. Therefore, to compute a product of non-disjoint cycles we need to proceed as follows. Take the example:

$$(a\ b\ c)(a\ b\ d)(c\ d\ b).$$

To compute this, we start with any element, say a, and compute its image. The element a is fixed by the rightmost cycle, then is mapped into b by the second cycle, and b is mapped into c by the leftmost cycle. So, the final destination of a is c. We then write:

$$(a\ b\ c)(a\ b\ d)(c\ d\ b) = (a\ c\ \ldots$$

and then proceed finding out the image of c: c goes to d, d goes to a, a goes to b, respectively, by the rightmost, middle, and leftmost cycle, so c is finally mapped into b. And so on. We reach the result:

$$(a\ b\ c)(a\ b\ d)(c\ d\ b) = (a\ c\ b)(d) = (a\ c\ b)$$

since singleton cycles do not need to be explicitly indicated.

One important operation is the *conjugation*. The conjugation of β by α is the permutation $\alpha\beta\alpha^{-1}$. This results in a permutation with the same cycle structure of β but the elements are changed by α. For instance, if $\beta = (b_1\ b_2\ \ldots\ b_k)$ then:

$$\alpha\beta\alpha^{-1} = (\alpha(b_1)\ \alpha(b_2)\ \ldots\ \alpha(b_k))$$

If β is a product of disjoint cycles, each one will be affected by α in the same way to form $\alpha\beta\alpha^{-1}$. Conjugations are so important that we will have a special notation for them: $\alpha\cdot\beta$ means the same as $\alpha\beta\alpha^{-1}$.

6.2.1 Short Cycles

A 2-cycle is a cycle of size 2. A 3-cycle is a cycle of order 3. It is important to know how products by 2- or 3-cycles affect an arbitrary permutation.

Let $\alpha = (a\ b)$ be a 2-cycle. Its effect on an arbitrary permutation β can be described as follows. If a and b are in the same cycle in β, this cycle is broken in two in $\alpha\beta$. If a and b are in two distinct cycles in β, these two cycles become one in $\alpha\beta$. Here and in the rest of the paper we say "cycle in β" meaning "cycle in the unique cycle decomposition of β".

The same results are valid for $\beta\alpha$. Notice that $\beta\alpha$ and $\alpha\beta$ are conjugates: $\alpha(\beta\alpha)\alpha^{-1} = \alpha\beta$, and therefore have the same cycle structure.

Now take an arbitrary 3-cycle $\alpha = (a\ b\ c)$ and an arbitrary permutation β. Three cases appear:

1. If a, b, and c are in three different cycles in β, these three cycles become a single cycle in $\alpha\beta$.

2. If two of a, b, c, are in the same cycle, and the third element is in a different cycle in β, then these two cycles recombine into another two cycles in $\alpha\beta$. Thus, the total number of cycles is maintained.

3. If a, b, and c are all in the same cycle in β, the result depends on the orientation they have in this cycle of β. Selecting a as the starting point, this cycle can have the form $(a \ldots b \ldots c \ldots)$ or $(a \ldots c \ldots b \ldots)$. In the first case, the cycle becomes $(a \ldots c \ldots b \ldots)$ in $\alpha\beta$. In the second case, the cycle breaks into $(a \ldots)(b \ldots)(c \ldots)$ in $\alpha\beta$.

The same results (except for the exact format of the resulting cycles in case 3) are valid for $\beta\alpha$.

6.2.2 Norm and Divisibility

In past sections we used phrases of the form "a, b, c in this order in π". It turns out that there is an algebraic way of expressing the same meaning. The notion of norm comes to our rescue.

It is well known that every permutation can be written as a product of 2-cycles. For instance, the cycle $(a\ b\ c\ d)$ can be written as:

$$(a\ b\ c\ d) = (a\ b)(b\ c)(c\ d)$$

This can be easily generalized to all cycles. Since every permutation is a product of cycles, it is clear that every permutation is a product of 2-cycles.

There are in general many ways of writing a permutation α as a product of 2-cycles. Among all these products, there must be some that use as few 2-cycles as possible. Let $|\alpha|$ denote the minimum number k such that α can written as a product of k 2-cycles. We call $|\alpha|$ the **norm** of α. For instance, $|1| = 0$, $|(a\ b)| = 1$ and $|(a\ b\ c\ d)| \leq 3$ (we will see later that this is an equality).

Theorem 6.2.1 *For every permutations α and β, he have:*

1. $|\alpha| = 0 \Leftrightarrow \alpha = 1$
2. $|\alpha^{-1}| = |\alpha|$
3. $|\beta \cdot \alpha| = |\alpha|$
4. $|\alpha\beta| \leq |\alpha| + |\beta|$

Proof: Item 1 is obvious, since using no cycles all we can produce is the identity. For item 2: if $\alpha = \alpha_1 \dots \alpha_k$, with each α_i being a 2-cycle, then $\alpha^{-1} = \alpha_k \dots \alpha_1$. So, for each decomposition of $|\alpha|$ into 2-cycles we have a decomposition of α^{-1} with the same number of cycles. This proves that $|\alpha^{-1}| \leq |\alpha|$. By symmetry we conclude that $|\alpha| \leq |\alpha^{-1}|$, and this proves item 2.

Also, if $\alpha = \alpha_1 \dots \alpha_k$, we have that:

$$\beta \cdot \alpha = (\beta \cdot \alpha_1)(\beta \cdot \alpha_2) \dots (\beta \cdot \alpha_k)$$

with each $\beta \cdot \alpha_i$ being a 2-cycle. Again, this allows us to conclude that $|\beta \cdot \alpha| \leq |\alpha|$. Since this is true for every α and β, we also have $|\beta^{-1} \cdot (\beta \cdot \alpha)| \leq |\beta \cdot \alpha|$ or, equivalently, $|\alpha| \leq |\beta \cdot \alpha|$. This proves item 3.

Item 4:

$$|\alpha| = k \Leftrightarrow \alpha = \alpha_1 \dots \alpha_k$$

$$|\beta| = l \Leftrightarrow \beta = \beta_1 \dots \beta_l$$

So,

$$\alpha\beta = \alpha_1 \dots \alpha_k \beta_1 \dots \beta_l \Rightarrow |\alpha\beta| \leq k + l = |\alpha| + |\beta|$$

This proves item 4.

■

Corollary 6.2.1 $|\alpha\beta| = |\beta\alpha|$

Let us start with the observations in Section 2.1. One of them says that if $\alpha = (a\ b)$, and a e b are in the same cycle in β, this cycle is broken in two in $\beta\alpha$. Calling $c(\alpha)$ the number of cycles in the cycle decomposition of a permutation α (including singleton cycles), this can be represented as a and b in the same cycle of $\beta \Rightarrow c(\beta\alpha) = c(\beta) + 1$. The other observation is: a and b in different cycles of $\beta \Rightarrow c(\beta\alpha) = c(\beta) - 1$.

These two facts are important because they say that the number of cycles changes by at most one when we multiply by a 2-cycle. Since the identity permutation in a set E has exactly $|E|$ cycles, we have the following result:

Theorem 6.2.2 *For any permutation α, if $\alpha = \alpha_1 \dots \alpha_k$, with each α_i being a 2-cycle, then $k \geq |E| - c(\alpha)$.*

Proof: Induction in k. For $k = 0$, we have $\alpha = 1$, $c(\alpha) = |E|$ and $|E| - c(\alpha) = 0$. For $k > 0$, let $\alpha' = \alpha_1 \dots \alpha_{k-1}$. We know by induction hypothesis that $k - 1 \geq |E| - c(\alpha')$. But $c(\alpha) = c(\alpha'\alpha_k) \geq c(\alpha') - 1$. Then $k = (k-1)+1 \geq |E| - c(\alpha') + 1 = |E| - (c(\alpha') - 1) \geq |E| - c(\alpha)$.
■

Corollary 6.2.2 *For any permutation α, we have $|\alpha| \geq |E| - c(\alpha)$.*

In fact, this is an equality, as the next result shows.

Theorem 6.2.3 *For any permutation α, we have $|\alpha| \leq |E| - c(\alpha)$.*

Proof: Let $\alpha = \alpha_1 \dots \alpha_k$ be the cycle decomposition of α including the 1-cycles as well. If cycle α_i has size l_i, we can write it as a product of 2-cycles with $l_i - 1$ terms (see beginning of Section 6.2.2). We therefore end up with a product of $\sum_{i=1}^{k}(l_i - 1)$ 2-cycles for α. But:

$$\sum_{i=1}^{k}(l_i - 1) = \sum_{i=1}^{k} l_i - k = |E| - c(\alpha)$$

Therefore, we have $|\alpha| \leq |E| - c(\alpha)$. ■

Corollary 6.2.3 *For any permutation α, we have $|\alpha| = |E| - c(\alpha)$.*

We say that a permutation α *divides* permutation β when $|\beta\alpha^{-1}| = |\beta| - |\alpha|$. We use the notation $\alpha|\beta$ to indicate this fact.

In the sequel, we present some basic propositions of the divisibility relation and later we use them to express algebraically the order of elements in a cycle.

Theorem 6.2.4 *The divisibility relation is an order relation, that is:*

1. $\alpha|\alpha$
2. $\alpha|\beta$ *and* $\beta|\alpha \Rightarrow \alpha = \beta$
3. $\alpha|\beta$ *and* $\beta|\delta \Rightarrow \alpha|\delta$

Proof:

1. $|\alpha\alpha^{-1}| = |1| = 0$ and $|\alpha| - |\alpha| = 0$.
2. We have that:

$$\alpha|\beta \Rightarrow |\beta\alpha^{-1}| = |\beta| - |\alpha|$$

$$\beta|\alpha \Rightarrow |\alpha\beta^{-1}| = |\alpha| - |\beta|$$

However, $\alpha\beta^{-1} = (\beta\alpha^{-1})^{-1}$, and therefore $|\beta\alpha^{-1}| = |\alpha\beta^{-1}|$. But then we have $|\beta\alpha^{-1}| = 0 \Rightarrow \beta\alpha^{-1} = 1 \Rightarrow \alpha = \beta$.

3. In this case, we have:

$$\alpha|\beta \Rightarrow |\beta\alpha^{-1}| = |\beta| - |\alpha|,$$
$$\beta|\delta \Rightarrow |\delta\beta^{-1}| = |\delta| - |\beta|.$$

So,

$$|\beta\alpha^{-1}| + |\delta\beta^{-1}| = |\delta| - |\alpha|.$$

But

$$|\delta\alpha^{-1}| = |\delta\beta^{-1}\beta\alpha^{-1}| \leq |\delta\beta^{-1}| + |\beta\alpha^{-1}| \leq |\delta| - |\alpha|,$$

and

$$|\delta| = |\delta\alpha^{-1}\alpha| \leq |\delta\alpha^{-1}| + |\alpha| \Rightarrow |\delta\alpha^{-1}| \geq |\delta| - |\alpha|$$

We conclude that $|\delta\alpha^{-1}| = |\delta| - |\alpha|$, that is, $\alpha|\delta$.

■

In addition, divisibility has some nice extra properties, and can be used to give an algebraic characterization of the fact that cycle α appears in the cycle decomposition of permutation β, a phrase that we usually abbreviate as "α is a cycle of β".

Theorem 6.2.5 *Let α be a cycle and β an arbitrary permutation. Then α is a cycle of β if and only if the following two conditions hold:*

1. $\alpha|\beta$
2. *there is no cycle α' with $\alpha|\alpha'|\beta$.*

Theorem 6.2.6 *If $\alpha|\beta$ then $\alpha^{-1}|\beta^{-1}$.*

We can use the divisibility relation to express concepts we use frequently, like the order of elements in a cycle or whether two elements belong to the same cycle or not in a permutation.

In view of the last corollary, we can say that $\alpha|\beta$ if and only if $c(\beta\alpha^{-1}) = c(\beta) + |\alpha|$, that is, if and only if the number of cycles in $\beta\alpha^{-1}$ increases by $|\alpha|$ compared to the number of cycles in β. In others words, α^{-1} causes maximum cycle breaking in β.

If α is the 2-cycle $(a\ b)$, α divides β if and only if a and b are in the same cycle of β. And if α is the 3-cycle $(a\ b\ c)$, α divides β if and only if a, b and c are in the same cycle of β, and appear in this order in it.

We summarize these observations in the following theorems.

Theorem 6.2.7 *Let $a, b, c \in E$ and α be a permutation. Then:*

1. *a, b are in the same cycle of $\alpha \iff (ab)|\alpha$*
2. *a, b, c in the same cycle of α and appear in this order in this cycle $\iff (abc)|\alpha$*

Theorem 6.2.8 *Let $\alpha = (ab), \beta$ be permutations. Then:*

1. *$\alpha|\beta \implies$ a cycle is broken in $\alpha\beta$ (when compared to β)*
2. *$\alpha \not| \beta \implies$ two cycles are joined in $\alpha\beta$ (when compared to β)*

Theorem 6.2.9 *Let $\alpha = (abc), \beta$ be permutations. Then:*

1. *$\alpha|\beta \iff$ a cycle is broken in three parts in $\beta\alpha^{-1}$ (when compared to β)*
2. *$\alpha \not| \beta$ but all 2-cycles (ab), (bc), (ac) divide $\beta \implies$ a cycle has two regions transposed in $\beta\alpha^{-1}$ (when compared to β)*
3. *$\alpha \not| \beta$ and exactly two of (ab), (bc), (ac) divide $\beta \implies$ two cycles are recombined in $\beta\alpha^{-1}$ (when compared to β)*
4. *(ab), (bc), (ac) do not divide $\beta \implies$ three cycles are joined in $\beta\alpha^{-1}$ (when compared to β)*

6.3 Genome Rearrangements

To formalize genome rearrangement problems we will use as base set for the permutations the set $E_n = \{-1, +1, -2, +2, \ldots, -n, +n\}$, where n is the number of genes. Thus, we will be modeling both strands of the underlying DNA molecule. Each element $+i$ or $-i$ represents a marker on the i^{th} gene, with its opposite meaning a marker in the same location in the opposite strand. We will first model circular genomes, which conform more naturally to the formalism, and will later comment on the necessary adaptations for linear genomes.

To begin with, let Γ be the permutation that maps each elements into its counterpart on the other strand. The permutation Γ can be written as:

$$\Gamma = (-1\ +1)(-2\ +2)\ldots(-n\ +n)$$

that is, a product of n disjoint 2-cycles. Notice that $\Gamma(a) \neq a$ for all $a \in E_n$, and $\Gamma^2(a) = \Gamma(\Gamma(a)) = a$ for all $a \in E_n$. In other words, $\Gamma^2 = 1$ or, equivalently, $\Gamma^{-1} = \Gamma$.

A cycle is *admissible* when it does not contain $-i$ and $+i$ for the same i. Thus, Γ is far from being an admissible cycle. An admissible cycle of size n is called a *genome strand*, because it models a strand of a genome formed by these n genes in some order. If we have an admissible cycle α, we can compute its reverse complement, as in the examples of the Table 6.1.

α	reverse complement
$(+3\ -1\ +7\ +5)$	$(-5\ -7\ +1\ -3)$
$(+2\ +4\ +6)$	$(-6\ -4\ -2)$

Table 6.1: Examples of admissible cycles α and their reverse complement.

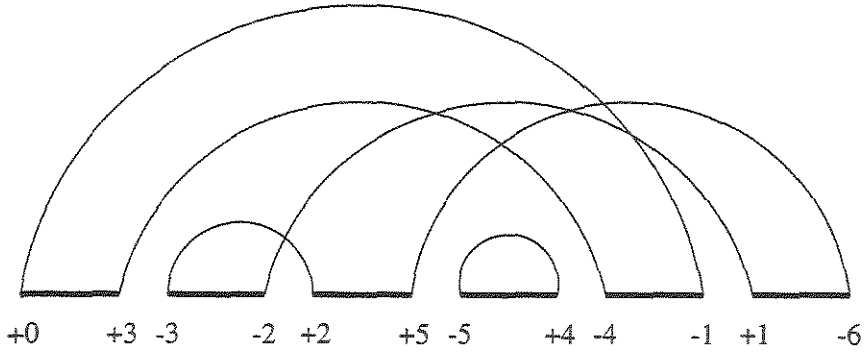

Figure 6.1: Breakpoint Graph for genomes $\pi = (-3\ +2\ -5\ -4\ +1)(-1\ +4\ +5\ -2\ +3)$ and $\sigma = (+1\ +2\ +3\ +4\ +5)(-5\ -4\ -3\ -2\ -1)$.

There is an algebraic way of obtaining the reverse complement. If α is an admissible cycle, α^{-1} is its reverse; $\Gamma \cdot \alpha = \Gamma\alpha\Gamma$ is its complement. The reverse complement is when we do both: $(\Gamma \cdot \alpha)^{-1}$ or $\Gamma \cdot (\alpha^{-1})$, which results in the same expression $\Gamma\alpha^{-1}\Gamma$.

Given a genome strand π_1, its reverse complement $\pi_2 = \Gamma\pi_1^{-1}\Gamma$ forms the complementary strand of the same genome. We represent this genome as the product of the two strands: $\pi = \pi_1\pi_2$. Since the strands form two disjoint cycles it does not matter in which order we take the product: $\pi_1\pi_2 = \pi_2\pi_1$. Also, it does not matter which strand we call π_1: had we started with π_2 we would have computed its reverse complement π_1 and the final genome would have been the same. This is just as DNA should be: no matter which strand you pick, when you let it pair with its reverse complement, you get the same DNA molecule.

Formally, we define a *genome* as a permutation that can be written as $\pi_1\Gamma\pi_1^{-1}\Gamma$, for some genome strand π_1. Note that $\Gamma\pi\Gamma = \pi^{-1}$ for every genome π. The general genome rearrangement problem then becomes: given two genomes π and σ and a class of operations, find the minimum number of events (operations) that transform π into σ. This minimum number is called the *distance* between π and σ.

We will talk about classes of operations later, but for any of the several problems obtained by choosing a different set of operations, the *breakpoint graph* plays an important role. Classically, the breakpoint graph is constructed as in Figure 6.1.

Details of the construction have been described previously several times and will not be repeated here. See for instance [64, 106]. Our objective is to obtain this graph, or an equivalent structure, by algebraic manipulations. The breakpoint graph is used when we want to transform

π into a constant genome $\sigma = (+1\ +2\ +3\ +4\ +5)(-5\ -4\ -3\ -2\ -1)$. It then depends on both π and σ. In fact, it has been defined for linear genomes, and to adapt to that we need consider "extended" versions of π and σ: take $\pi_1 = (+0\ -3\ +2\ -5\ -4\ +1)$ and identify -0 with -6.

The breakpoint graph is composed of black edges, which depend only on π, and of gray edges, which depend only on σ. Its turns out that $\Gamma\pi$ is a product of 2-cycles that correspond exactly to the black edges. And $\Gamma\sigma$ corresponds to the gray edges in the same way. In the preceding example, we have:

$$\begin{aligned}\Gamma\pi &= (-0\ +0)(-1\ +1)(-2\ +2)(-3\ +3)(-4\ +4)(-5\ +5)\\ &\quad (+0\ -3\ +2\ -5\ -4\ +1)(-1\ +4\ +5\ -2\ +3\ -0)\\ &= (+0\ +3)(-3\ -2)(+2\ +5)(-5\ +4)(-4\ -1)(+1\ -0),\end{aligned}$$

exactly the black edges. And $\Gamma\sigma$ will give the gray edges:

$$\Gamma\sigma = (+0\ -1)(+1\ -2)(+2\ -3)(+3\ -4)(+4\ -5)(+5\ -0).$$

In the classical theory of genome rearrangements the cycle structure of the breakpoint graph plays an important role. Although we could not obtain the cycles of the breakpoint graph themselves, we derived an algebraic expression for the *square* of the cycles. This expression is just the product $(\Gamma\pi)(\Gamma\sigma) = \Gamma\pi\Gamma\sigma$. In the example, we have:

$$\Gamma\pi\Gamma\sigma = (+0\ -4)(+3\ -1)(-3\ +5\ +1)(-2\ -0\ +2)(-5)(+4)$$

For each cycle of the breakpoint graph we have two cycles in $\Gamma\pi\Gamma\sigma$. If the cycle in the breakpoint graph is $(a_1\ a_2\ \ldots\ a_{2k})$, we have $(a_1\ a_3\ \ldots\ a_{2k-1})$ and $(a_{2k}\ a_{2k-2}\ \ldots\ a_2)$ in $\Gamma\pi\Gamma\sigma$. Therefore, this is not exactly the square of a breakpoint cycle, because one of them is reversed. Strictly speaking, we cannot model as permutations the cycles of the breakpoint graph, since they have no orientation. This in part explains why one cycle in the square is reversed. Had we taken $\Gamma\pi\Gamma\sigma$ the other cycle would have been reversed. Notice that $\Gamma\pi\Gamma\sigma = \pi^{-1}\sigma$, and $\Gamma\sigma\Gamma\pi = \sigma^{-1}\pi$.

In any case, these constructions allow us to rephrase technical properties of breakpoint graphs in algebraic terms. For instance, how many breakpoints π has with respect to σ? This is just the number of elements not fixed by $\Gamma\pi\Gamma\sigma$, divided by 2:

$$b(\pi,\sigma) = \frac{|Supp(\Gamma\pi\Gamma\sigma)|}{2}.$$

Likewise, the number of cycles in the breakpoint graph is half the number of cycles in $\pi^{-1}\sigma$. We can define also the length (size) of the cycles, good or bad cycles, cycles that can be broken by certain operations. We hope to be able to define interleaving cycles, hurdles, fortresses, all in algebraic terms.

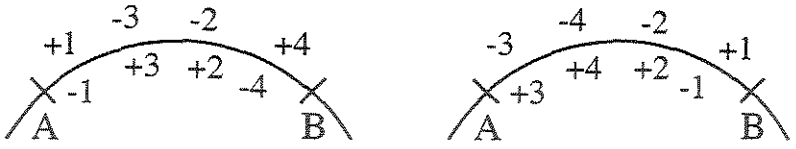

Figure 6.2: Linear genomes with fixed extremes.

Figure 6.3: Linear genomes without fixed extremes.

6.3.1 Linear Genomes

The theory developed so far fits nicely with circular genomes. In this section we will briefly examine the case of linear genomes.

First, we must recognize that there are actually two kinds of linear genome: with free and with fixed extremes. Let us define each kind, starting with the one with fixed extremes.

When we compare two regions of two different genomes, and these regions are flanked by conserved parts, we need to use the fixed-extreme case (Figure 6.2). In this case, we add an extra dummy gene BA, which represents the fixed extremities of the regions, and proceed as in the circular case.

When we compare two entire linear genomes, we need to take into account that there is a free reversal that can be applied, so the distance in this case becomes:

$$d_{free} = \min(d_{fixed}(\pi, \sigma), d_{fixed}(\Gamma \cdot \pi^{-1}, \sigma))$$

More details on the relationship between linear and circular genome rearrangement problems can be found in the references [119, 92].

6.3.2 Operations

We will define in this section the events (operations) of reversal, transposition (or block move), and block interchange, some of them in the their signed and unsigned version. We will do a very detailed job for reversals, and then just state the results for the others, to save space.

Given a genome π, to perform a reversal on it we need to choose two distinct markers u and v, in the same strand of π, and then replace the path from u to v (including u but excluding v) by its reverse complement. Of course, a similar operation will be performed on the other strand, to make sure the final result is still a valid genome. Figure 6.4 shows what is meant.

We want to write the resulting genome σ as $\rho\pi$, where ρ is a permutation that will represent

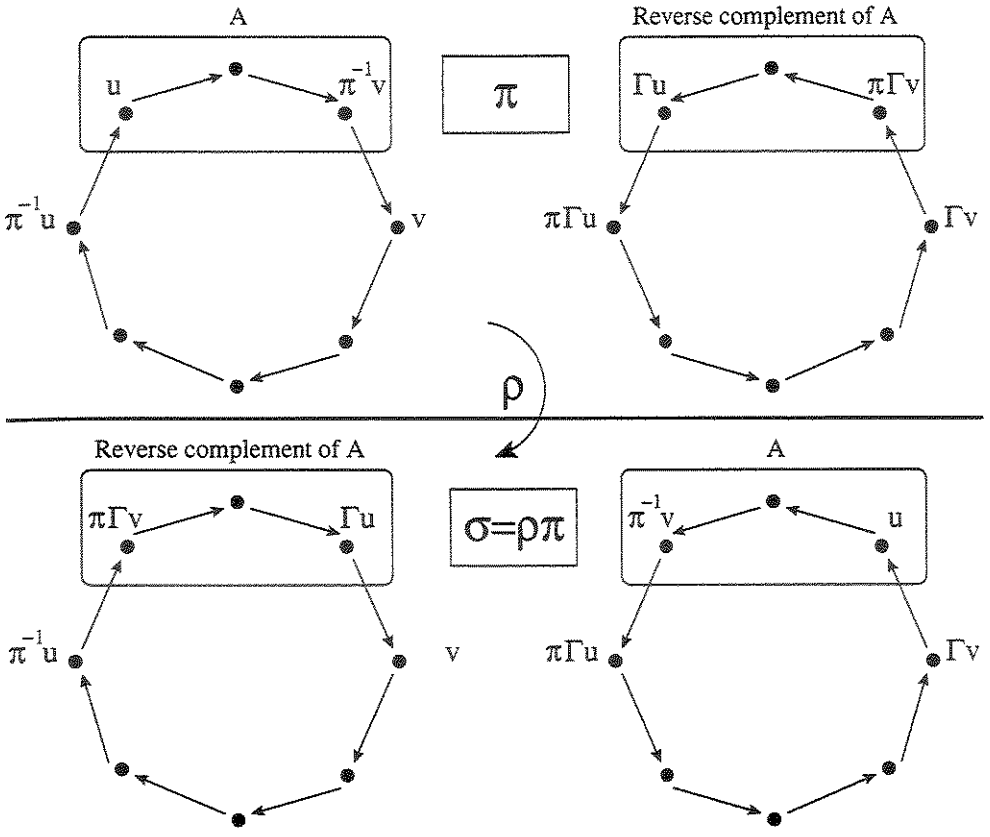

Figure 6.4: A reversal ρ applied to genome π.

the reversal. With some work, we see that σ differs from π only in the following mappings:

$$\sigma\pi^{-1}u = \pi\Gamma v,\ \sigma\Gamma u = v,\ \sigma\Gamma v = u,\ \sigma\pi^{-1}v = \pi\Gamma u.$$

Therefore, $\rho = \sigma\pi^{-1}$ maps:

$$\rho\pi\Gamma v = \sigma\pi^{-1}\pi\Gamma v = \sigma\Gamma v = u$$

$$\rho v = \sigma\pi^{-1}v = \pi\Gamma u$$

$$\rho u = \sigma\pi^{-1}u = \pi\Gamma v$$

$$\rho\pi\Gamma u = \sigma\pi^{-1}\pi\Gamma u = \sigma\Gamma u = v$$

with all other elements fixed by ρ. Or, written as a product of disjoint cycles:

$$\rho = (u\ \pi\Gamma v)(v\ \pi\Gamma u).$$

This is then the general formula of a reversal applicable to π, where u and v are two elements in same strand of π. We say that π and σ *differ by a reversal* when there is such a reversal ρ

with $\sigma = \rho\pi$. Notice that the definition of a reversal depends on π. There is no way to define a class of permutations that will be "the reversals", valid for all genomes. Each genome has a particular set of reversals that can be applied to it, and this sets varies from one genome to another.

For this reason, we cannot view the genome rearrangement problem directly as a "group generators" problem, where a class of generators of the symmetric group is given and we seek the minimum number of generators to write a given permutation. Nevertheless, it can be viewed indirectly as a group generators problem. We intend to pursue this approach in a future paper.

The reversal distance problem is: given two genomes π and σ, find the minimum k such that there are genomes $\gamma_0, \gamma_1, \ldots, \gamma_k$ with $\pi = \gamma_0, \sigma = \gamma_k$ and γ_i differs from γ_{i+1} by a reversal, for $i = 0, \ldots, (k-1)$.

In a previous version of this manuscript we stated that unsigned reversals could also be handled, but this is false, as far as we can see, because unsigned reversals do not exist in nature. They are artifacts created to deal with the lack of information on orientation. However, the following operations can be handled by this formalism.

A *transposition* is defined as:

$$\tau = (\pi u\ \pi v\ \pi w)(\Gamma w\ \Gamma v\ \Gamma u)$$

where u, v and w are three distinct elements in the same strand in π, appearing in this order (u, v, w) in the strand.

A *reversal+transposition* is defined as:

$$\tau = (\pi u\ \pi v\ \Gamma w)(\pi w\ \Gamma v\ \Gamma u)$$

where u, v and w are distinct elements in the same strand in π, appearing in this order (u, v, w) in this strand. A reversal+transposition models the event in which a block detaches itself from a genome and reappears elsewhere, in the same strand but the block is reversed.

A *block interchange* is defined as:

$$\beta = (\pi u\ \pi w)(\pi x\ \pi v)(\Gamma u\ \Gamma w)(\Gamma x\ \Gamma v)$$

where u, v, w and x are four distinct elements in the same strand of π, appearing in the order (u, v, w, x) in this strand.

Each one (or a group of) of these types of events can be used to define a genome rearrangement problem: given two genomes π and σ, find the minimum k such that there are genomes $\gamma_0, \gamma_1, \ldots, \gamma_k$ with $\pi = \gamma_0$, $\sigma = \gamma_k$ and γ_i differs from γ_{i+1} by the specific operation (or a group of), for $i = 0, \ldots, (k-1)$.

6.4 Using the Theory

We will use the theory developed to show two results whose proof was based on pictorial representations. The first result appears in Christie's proof that a block interchange cannot create three cycles [28]. The other result is that there is only one way for a transposition to break a cycle, proved by Walter & colleagues by reference to a picture [120].

Theorem 6.4.1 *Let π and σ be two genomes, and β a block interchange on π. Then the number of cycles in $\sigma(\beta\pi)^{-1}$ is not higher than 4 plus the number of cycles in $\sigma\pi^{-1}$.*

Proof: We have $\sigma(\beta\pi)^{-1} = \sigma\pi^{-1}\beta^{-1}$, which is $\sigma\pi^{-1}$ multiplied by β^{-1}. The cycle structure of β^{-1} is the same as β's: four 2-cycles. By the classical results about products by 2-cycles it is immediate that multiplying by four 2-cycles we cannot create more than 4 extra cycles. ■

Theorem 6.4.2 *Let π and σ be two genomes, and $\tau = (u\ v\ w)(\pi\Gamma w\ \pi\Gamma v\ \pi\Gamma u)$ a transposition on π, where u, v, w appear in this order in the same cycle of π. Then $\sigma(\tau\pi)^{-1}$ has four more cycles than $\sigma\pi^{-1}$ if and only if u, v, w are in the same cycle in $\sigma\pi^{-1}$ and appear in the order (u, v, w) in this cycle.*

Proof: Transpositions do not mix genome strands, and therefore we know that the elements of a strand of π will form a strand in $\tau\pi$ (possibly in different order). Let π_1 be the strand that contains u, v and w, and σ_1, the corresponding strand in σ. We then have $\pi = \pi_1\Gamma\pi_1^{-1}\Gamma$, $\sigma = \sigma_1\Gamma\sigma_1^{-1}\Gamma$, and $\sigma\pi^{-1} = \sigma_1\pi_1^{-1}(\Gamma\sigma_1^{-1}\Gamma)(\Gamma\pi_1\Gamma)$. Then $\sigma(\tau\pi)^{-1} = \sigma\pi^{-1}\tau^{-1}$ will be the product of disjoint permutations $\sigma_1\pi_1^{-1}(w\ v\ u)$, and $(\Gamma\sigma_1^{-1}\Gamma)(\Gamma\pi_1\Gamma)(\pi\Gamma u\ \pi\Gamma v\ \pi\Gamma w)$.

In the first component $\sigma_1\pi_1^{-1}(w\ v\ u)$ we have a product of a 3-cycle by $\sigma_1\pi_1^{-1}$. We know from Theorem 6.2.9) that this produces two extra cycles if and only if $(uvw)|\sigma_1\pi_1^{-1}$, that is, if and only if u, v, w appear in the same cycle of $\sigma_1\pi_1^{-1}$ in the order (u, v, w), as stated. ■

6.5 Reversal Distance

In this section we will apply our theory to the reversal distance problem.

6.5.1 Good and Bad Cycle Pairs

Let us start by examining the cycle structure of $\sigma\pi^{-1}$, where σ and π are two genomes. It turns out that every cycle α in $\sigma\pi^{-1}$ has a *companion* cycle $(\pi\Gamma)\cdot\alpha^{-1}$ also in $\sigma\pi^{-1}$. Their product will be called a *cycle pair* in $\sigma\pi^{-1}$

Every cycle α in $\pi^{-1}\sigma$ has a companion cycle $(\Gamma\pi)\cdot\alpha^{-1}$ also in $\pi^{-1}\sigma$. The cycle α is good if and only if $(\Gamma\pi)\cdot\alpha^{-1}$ is good.

First, we note that there are four ways of specifying the pivot points u, v of a reversal $\rho(u, v)$, because:

$$\rho(u, v) = \rho(v, u) = \rho(\pi\Gamma u, \pi\Gamma v) = \rho(\pi\Gamma v, \pi\Gamma u).$$

According to the relative positions of u, v in $\sigma\pi^{-1}$ we have three cases:

1. u, v in the same cycle of $\sigma\pi^{-1}$
2. u, v in companion cycles of $\sigma\pi^{-1}$
3. u, v in non-companion cycles of $\sigma\pi^{-1}$

Notice that any of these conditions, if true for u and v, is also true for v, u, for $\pi\Gamma u, \pi\Gamma v$, and for $\pi\Gamma v, \pi\Gamma u$, that is, they are independent of the particular pivot points chosen and are therefore characteristics of the reversal itself.

In case 1 the number of cycles of $\sigma\pi^{-1}$ does not change, because we have:

$$\sigma\pi^{-1} = \alpha(u \ldots v \ldots)(\ldots \pi\Gamma v \ldots \pi\Gamma u)$$

Then,

$$\begin{aligned} \sigma\pi^{-1}\rho^{-1} &= \sigma\pi^{-1}\rho \\ &= \alpha(u \ldots v \ldots)(\ldots \pi\Gamma v \ldots \pi\Gamma u)(u\ \pi\Gamma v)(v\ \pi\Gamma u) \\ &= \alpha(u \ldots \pi\Gamma u \ldots)(v \ldots \pi\Gamma v \ldots) \end{aligned}$$

In case 2 the number of cycles of $\sigma\pi^{-1}$ increases by 2:

$$\sigma\pi^{-1} = \alpha(u \ldots \pi\Gamma v \ldots)(\ldots v \ldots \pi\Gamma u)$$

Then,

$$\begin{aligned} \sigma\pi^{-1}\rho^{-1} &= \sigma\pi^{-1}\rho \\ &= \alpha(u \ldots \pi\Gamma v \ldots)(\ldots v \ldots \pi\Gamma u)(u\ \pi\Gamma v)(v\ \pi\Gamma u) \\ &= \alpha(u \ldots)(v \ldots)(\pi\Gamma u \ldots)(\pi\Gamma v \ldots) \end{aligned}$$

In case 3 the number of cycles of $\sigma\pi^{-1}$ decreases by 2:

$$\sigma\pi^{-1} = \alpha(u \ldots)(\ldots \pi\Gamma u)(v \ldots)(\ldots \pi\Gamma v)$$

Then,

$$\begin{aligned} \sigma\pi^{-1}\rho^{-1} &= \sigma\pi^{-1}\rho \\ &= \alpha(u \ldots)(\ldots \pi\Gamma u)(v \ldots)(\ldots \pi\Gamma v)(u\ \pi\Gamma v)(v\ \pi\Gamma u) \\ &= \alpha(u \ldots \pi\Gamma v \ldots)(v \ldots \pi\Gamma u \ldots) \end{aligned}$$

Based on these observations, we say that a reversal $\rho(u, v)$ that can be applied to genome π is a *cycle twister* in case 1, a *cycle breaker* in case 2, and a *cycle merger* in case 3. In the first two cases we say that $\rho(u, v)$ *acts on the cycle pair* that contains u and v. Because there are alternative ways of picking the pivot points for a given reversal, it is necessary to refer to the cycle pair as a whole and not just to the cycle individually.

Take an arbitrary cycle α of $\sigma\pi^{-1}$. Let k be the length of this cycle, and let k_1 and k_2 be the number of elements in this cycle that are in strands π_1 and π_2 of π. Define the *imbalance* of α as $|k_1 - k_2|$. This number has the same parity as k and lies between 0 and k. It is in fact a property of the cycle pair of α, because the companion cycle $(\pi\Gamma) \cdot \alpha$ will have the same imbalance.

We define a **bad** $\sigma\pi^{-1}$ cycle pair as a cycle pair that has imbalance equal to k, where k is the length of its cycles. A **good** $\sigma\pi^{-1}$ cycle pair is a cycle pair with imbalance less than k.

Theorem 6.5.1 *A cycle pair γ is good if and only if there is a cycle breaker that acts on γ.*

Proof: If γ is good then there are u, v in the same cycle of γ and in different strands of π. The reversal $\rho(u, \pi\Gamma v)$ is a cycle breaker acting on γ.

Conversely, let $\rho(u, v)$ be a cycle breaker of γ. Then u, v are in companion cycles. But then u, $\pi\Gamma v$ are in the same cycle of γ and in different strands of π. ■

6.5.2 Interleaving Cycles

Two cycle pairs γ_1 and γ_2 **interleave** when there are reversals ρ_1 acting on γ_1 and ρ_2 acting on γ_2 such that $\rho_1\rho_2\pi \neq \rho_2\rho_1\pi$. We can construct a graph where each cycle pair is a vertex, and interleaving cycle pairs are joined by an edge. A connected component of this graph is *good* when there is at least one good cycle pair in it. Otherwise it is a *bad* component.

Theorem 6.5.2 *If all components are good, then*

$$d(\pi, \sigma) = \frac{|\sigma\pi^{-1}|}{2}.$$

6.6 Conclusions

We propose a new way of looking of genomes as permutations, one that is more comfortable for those that have experience in permutation groups. Much remains do be done, but we feel this is the right way to attack difficult problems.

Capítulo 7

The Genome Distance Problem by Fusion, Fission, and Transposition is Easy *

Zanoni Dias

João Meidanis

Abstract

Given two genomes represented as circularly ordered sequences of genes, we show a polynomial time algorithm for the minimum weight series of fusion, fissions, and transpositions (with transpositions weighing twice as much as fusions and fissions) that transforms one genome into the other. The algorithm is based on classical results of permutation group theory and is the first polynomial result for a genome rearrangement problem involving transpositions. It has been observed in real biological instances that transpositions occur with about half the frequency of reversals. Although we are not using reversals in this study, this observation motivated the double weight assigned to transpositions.

Trabalho depositado como Relatório Técnico no Instituto de Computação da Unicamp em julho de 2001, sob o número IC-01-07. Este artigo foi apresentado no **String Processing and Information Retrieval** *(SPIRE'2001) realizado na Laguna de San Rafael, no Chile, em novembro de 2001.*

7.1 Introduction

With the advent of fast sequencing techniques, we are witnessing today a spectacular increase in the quantity of molecular data (DNA and protein sequences). More than 40 complete microbial genomes are known now, and about 170 others are in progress [114]. The great challenge we face now is how to process this huge amount of data and extract from it relevant biological information that could help design drugs, understand life and disease, improve crops, and so on. One way to structure this information is by comparative genomics, where we analyze data coming from distinct species and learn from the similarities and differences in related genomes. Among the several proposed ways of comparing genomes, the area of **genome rearrangements** has received a lot of attention recently [8, 7, 61, 64, 62, 55, 92, 117, 28, 30, 119]. In this area, very large DNA molecules (usually entire chromosomes or large pieces of chromosomes) are investigated with respect to the relative order of genes in them. The goal is to determine a rearrangement distance, which is the minimum number of rearrangement events that could explain the differences between two such DNA molecules.

Many different events have been considered. Reversals, transpositions and translocations are the best studied ones from a theoretical point of view, although in practice events such as duplications and deletions are at least as important. As far reversals are concerned, Hannenhalli and Pevzner presented the first polynomial time algorithm [64], subsequently improved by Kaplan, Shamir, and Tarjan [73]. Caprara showed that the reversal problem is NP-hard if we disregard the orientation of genes [21]. Hannenhalli and Pevzner also solved in polynomial time a multi-chromosomal problem involving translocation, fusion and fission [62]. Bafna and Pevzner [9] studied the transposition distance between two linear unsigned chromosomes, presenting several approximation algorithms, the best one having approximation factor 1.5 and running in $O(n^2)$ time, and suggesting some open problems. Christie [30] devised an alternative 1.5-approximation algorithm that runs in $O(n^4)$ time. Guyer, Heath, and Vergara [58] implemented several algorithms for computing the transposition distance, based on subsequences and runs in a permutation. Christie [28] proposed and solved the problem of block-interchange distance. A block-interchange can be viewed as a generalization of a transposition. In a block-interchange two non-intersecting substrings of any length are swapped in the permutation. In a transposition the substrings must be adjacent. Transposition distance seems to be a harder problem, that has eluded researchers for many years now. Its computational complexity is still unknown.

We show a polynomial time algorithm for the minimum weight series of fusion, fissions, and transpositions (with transpositions weighing twice as much as fusions and fissions) that transforms one genome into the other. The algorithm is based on classic results of permutation group theory and it is the first polynomial result for a genome rearrangement problem involving transpositions. It has been observed [16] in real biological instances that transpositions occur

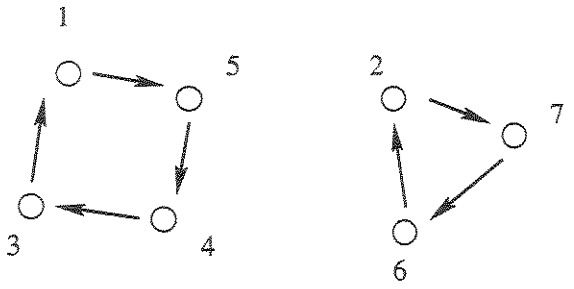

Figure 7.1: A multi-chromosomal genome.

with about half the frequency of reversals.

In the following sections we present definitions, the main result of this work, proof sketches and conclusions and plans for future work.

7.2 Definitions, Modeling, and Results

A **permutation** in group theory is a one-to-one mapping from a set E into itself. We will use permutations to represent genomes with circular chromosomes. The standard notation [87] for permutations is to represent in parenthesis an element followed by its successive images. For instance, if $E = \{1, 2, 3, 4, 5\}$ the permutation α such that

$$\alpha(1) = 3, \alpha(2) = 5, \alpha(3) = 2, \alpha(4) = 1, \alpha(5) = 4$$

is represented as

$$(1\ 3\ 2\ 5\ 4).$$

The representation is not unique since we could have started at an element other than 1: (2 5 4 1 3), (5 4 1 3 2), etc. are all equivalent.

In our model, the set E is the set of genes of the genome and the permutation indicates how genes follow each other in the chromosomes. Only circular chromosomes can be represented in this way, but an easy translation of results from circular to linear chromosomes exists [90].

Permutations represent also multi-chromosomal genomes. For instance, if $E = \{1, 2, 3, 4, 5, 6, 7\}$ the permutation (1 5 4 3)(2 7 6) represents the genome depicted in Figure 7.1.

An element x if fixed under a permutation α when $\alpha(x) = x$. Fixed elements can be omitted in the parenthesized notation for permutations. For instance, if α is such that

$$\alpha(1) = 1, \alpha(2) = 3, \alpha(3) = 4, \alpha(4) = 2, \alpha(5) = 5,$$

then we can write $\alpha = (2\ 3\ 4)$, or $\alpha = (2\ 3\ 4)(1)$ or $\alpha = (1)(2\ 3\ 4)(5)$. The missing elements are implicitly understood as fixed. The **support** of a permutation α is the set of elements not fixed by α. In the preceding example, we have $Supp(\alpha) = \{2, 3, 4\}$. The identity permutation, denoted simply by 1 (without parenthesis), fixes all elements and has empty support.

Permutations can be composed as mappings. This defines a product of permutations. For instance, if $E = \{1, 2, 3, 4, 5, 6\}$, $\alpha = (2\ 3\ 4)$ and $\beta = (3\ 1\ 5\ 2\ 6\ 4)$ we have $\alpha\beta$ defined as $\alpha\beta(x) = \alpha(\beta(x))$ for all $x \in E$, and therefore $\alpha\beta = (1\ 5\ 3)(2\ 6)$. Any two permutations over the same set can composed in this way. The operation is associative, the identity permutation is the identity element, and every permutation α has an inverse α^{-1} [87].

Composition of permutations is important in the context of genome rearrangements for at least two reasons:

- Some permutations ρ with small support can be viewed as rearrangement events: $\rho\pi$ is then the result of event ρ acting on genome π.
- Given two genomes σ and π, the product $\sigma\pi^{-1}$describes in some sense the "differences" between the two genomes.

For instance, fusions and fissions can be seen as permutations of support size 2, and transpositions can be seen as permutations of support size 3, as we will see shortly. In addition, our main result establishes that the distance between two genomes can be computed as n minus the number of cycles in $\sigma\pi^{-1}$ (see Section 7.2.1 for a formal definition of cycles).

7.2.1 Orbits and Cycles

Any permutation can be written in a unique way as the product of cycles disjoint support (disjoint cycles). To understand cycles we need first the definition of orbit. An orbit can be defined intuitively as a set of the form $x, \alpha(x), \alpha^2(x), \ldots$ for some element x. Since we are dealing with finite sets, orbits are always finite, that is, there is a positive integer k such that $\alpha^k(x) = x$. A more formal definition is given below.

Definition 7.2.1 *An **orbit** of a permutation α is a minimal set of elements A such that for any two elements $x, y \in A$ there is an integer k such that $\alpha^k(x) = y$.*

For instance, if $E = \{1, 2, 3, 4, 5, 6\}$ and $\alpha = (2\ 3\ 5)(1\ 4)$ then the orbits are $\{2, 3, 5\}$, $\{1, 4\}$, and $\{6\}$. Restricting α to one of its orbits we obtain what is called a cycle of α. Formally, we have the following definition.

Definition 7.2.2 *A **cycle** of a permutation α is a permutation β such that there is an orbit A of α with*

$$\beta(x) = \begin{cases} \alpha(x) & \text{if } x \in A \\ x & \text{if } x \notin A \end{cases}$$

For instance, if $E = \{1, 2, 3, 4, 5, 6\}$ and $\alpha = (2\ 3\ 5)(1\ 4)$ then the cycles of α are $(2\ 3\ 5)$, $(1\ 4)$, and (6). A permutation that is a cycle of some permutation is called simply a **cycle**. The **size** of the cycle is the number of elements of its non-singleton orbit if there is one, or 1 if all orbits are singletons. A cycle of size k is also called a **k-cycle**.

It is important to note here a potentially confusion terminology used in the literature. The term "transposition" is used in permutation group theory meaning 2-cycle. The same term "transposition" is used in biology meaning a block move in a genome. Unfortunately, in both areas the term "transposition" is well-established and very unlikely to change. We had to make a choice in this paper, and decided to keep the biological meaning. Whenever we need to refer to the group theoretical meaning of "transposition" we will use 2-cycle instead.

7.2.2 Rearrangement Events

We will define now the rearrangement events that are the main subject of this paper: fusion, fission, and transposition. Intuitively, a fusion joins two chromosomes into one; a fission breaks a chromosomes into two; and a transposition moves a block of consecutive genes from our place into another in the same chromosome.

Formally, fusions and fissions correspond to 2-cycles. Given a 2-cycle $\rho = (x\ y)$ and a permutation π, we have the following classical results:

- if x and y are in the same cycle of π, then in $\rho\pi$ this cycle is broken into two cycles, one containing x and the other y (among others elements). The remaining cycles of π are left unchanged.
- if x and y are in the distinct cycle of π, then in $\rho\pi$ these two cycles are joined into one. The remaining cycles of π are left unchanged.

These classical results show that 2-cycles correctly model fusions and fissions, because the cycles of a permutation correspond to circular chromosomes of a genome. However, notice that $\rho = (x\ y)$ can act as a fusion for some genomes and as a fission for others. Therefore, being a fusion (or a fission) is not an intrinsic property of ρ but rather depends also on the genome π on which ρ is being applied. Nevertheless, all fusions and fissions are captured by 2-cycles.

Formally, transpositions correspond to 3-cycles. Again, a 3-cycle is not intrinsically a transposition, but rather its transposition status depends on the particular genome on which it is being applied. More specifically, a 3-cycle $\rho = (x\ y\ z)$ is a transposition when it acts on a genome π where the elements x,y,z are all in the same cycle and appear in this order in the cycle. For instance, if $\rho = (7\ 3\ 2)$ and $\pi = (7\ 1\ 5\ 3\ 2\ 6\ 4)$ then $\rho\pi = (7\ 1\ 5\ 2\ 6\ 4\ 3)$ and ρ models a transposition.

We are now ready to define our problem formally. Given two genomes represented by permutations π and σ over the same set of genes, find a series of events $\rho_1, \rho_2, \ldots, \rho_k$ such that:

- $\rho_k \rho_{k-1} \cdots \rho_2 \rho_1 \pi = \sigma$
- each ρ_i is a fusion, fission or transposition for the genome $\rho_{i-1}\rho_{i-2} \cdots \rho_2 \rho_1 \pi$ on which it acts, and
- $\Sigma_{i=1}^{k} w(\rho_i)$ is minimum, where $w(\rho_i)$ is 1 if ρ_i is a fusion or fission, and 2 if ρ_i is a transposition.

The minimum value of $\Sigma_{i=1}^{k} w(\rho_i)$ is called the distance between π and σ.

The main result of this paper is that the problem just defined is solvable in polynomial time, This follows from a classical result in permutation group theory as we will see in the next section.

7.3 Proof Sketches

In this section we will sketch some of the proofs need in our main result. We begin with some additional definitions, continue with a formal statement of the main result, and finish with the proof sketches.

Definition 7.3.1 *A permutation ρ is a **valid event** for another permutation (genome) π with either:*

1. *ρ is a 2-cycle, or*
2. *ρ is a transposition when applied to π.*

*In case (1) the **weight** of ρ, denoted by $w(\rho)$, is equal to 1; in case (2), $w(\rho) = 2$.*

Notice that if ρ is a valid event for π then ρ^{-1} is a valid event for $\rho\pi$. In other words, valid events can be "undone" by other valid events of the same weight.

Definition 7.3.2 *An ordered sequence of permutations (ρ_1, ρ_2, . . ., ρ_k) is a **series of valid events leading from π to σ** when:*

- *each ρ_i is a valid event for $\rho_{i-1}\rho_{i-2} \cdots \rho_2 \rho_1 \pi$, and*
- $\rho_k \rho_{k-1} \cdots \rho_2 \rho_1 \pi = \sigma$

We are interested in such a series with minimum total weight $\Sigma_{i=1}^{k} w(\rho_i)$.

Definition 7.3.3 *For a permutation α, let $c(\alpha)$ denote the number of orbits of α. For two permutations (genomes) π and σ, let $c(\pi, \sigma)$ denote the number of orbits of $\sigma\pi^{-1}$.*

For instance, $c(\pi, \pi) = n$ for any genome π, where $n = |E|$ is the number of genes.

Definition 7.3.4 *Given two permutations (genomes) π and σ and a valid event ρ for π, denote by $\Delta c(\rho, \pi, \sigma)$ the value:*

$$\Delta c(\rho, \pi, \sigma) = c(\rho\pi, \sigma) - c(\pi, \sigma)$$

The quantity $\Delta c(\rho, \pi, \sigma)$ is the increase in the number of orbits of $\sigma\pi^{-1}$ when π is replaced by $\rho\pi$. If this number is positive, $\rho\pi$ is "closer" to σ then π was.

Definition 7.3.5 *A valid event ρ for π is **good with respect to** σ when:*

$$\Delta c(\rho, \pi, \sigma) = w(\rho)$$

Our main result can be stated as follows.

Theorem 7.3.1 *Given two permutations (genomes) π and σ, the distance between them is $n - c(\pi, \sigma)$.*

The proof relies on the following two lemmas.

Lemma 7.3.1 *For any series of valid events (ρ_1, ρ_2, ..., ρ_k) leading from π to σ we have:*

$$\Sigma_{i=1}^{k} w(\rho_i) \geq n - c(\pi, \sigma)$$

with equality if and only if each ρ_i is good for $\rho_{i-1}\rho_{i-2}\ldots\rho_2\rho_1\pi$ with respect to σ.

Proof: Suppose that

$$\rho_k\rho_{k-1}\ldots\rho_2\rho_1\pi = \sigma \tag{7.1}$$

Each ρ_i is a 2-cycle or a 3-cycle, but any 3-cycle can be written as a product of two 2-cycles. Replacing every 3-cycle of equation (7.1) by a product of 2-cycles, we have:

$$c_{k'}c_{k'-1}\ldots c_2c_1\pi = \sigma\pi^{-1}$$

where each c_i is a 2-cycle. The number of 2-cycles involved is just:

$$k' = \Sigma_{i=1}^{k} w(\rho_i)$$

since 2-cycles have weight 1 and 3-cycles weight 2. But there is a classical result that says that if a permutation α can be written as a product of 2-cycles, the number of 2-cycles is at least $n - c(\alpha)$ [87].

Therefore,

$$k' \geq n - c(\alpha)$$

or

$$\Sigma_{i=1}^{k} w(\rho_i) \geq n - c(\pi, \sigma)$$

■

Lemma 7.3.2 *Given two distinct permutations (genomes) π and σ, there is always a good event for π with respect to σ.*

Proof: Since $\pi \neq \sigma$ we have $\sigma\pi^{-1} \neq 1$ and there is a k-cycle in $\sigma\pi^{-1}$ with $k \geq 2$. Choose x and y as two distinct elements in this k-cycle. We claim that the event $\rho = (x\ y)$ is valid for π and is a good event with respect to σ.

The event ρ is valid for π since it is a 2-cycle, and therefore is either a fusion or a fission in π. It is a good event with respect to σ because of the following argument. By choice we know that ρ splits a cycle of $\sigma\pi^{-1}$ into two. Therefore,

$$c(\sigma\pi^{-1}\rho) = c(\sigma\pi^{-1}) + 1$$

In addition, ρ is a 2-cycle so $\rho = \rho^{-1}$. But then:

$$\begin{aligned} \Delta(\rho, \pi, \sigma) &= c(\rho\pi, \sigma) - c(\pi, \sigma) = c(\sigma\pi^{-1}\rho^{-1}) - c(\sigma\pi^{-1}) \\ &= c(\sigma\pi^{-1}) + 1 - c(\sigma\pi^{-1}) = 1 = w(\rho) \end{aligned}$$

■

This suggests the following algorithm for finding the distance and an optimal series of events leading from π to σ.

FUSION, FISSION, AND TRANSPOSITION DISTANCE()
1 Input π, σ
2 $d \leftarrow 0$
3 **While** $\pi \neq \sigma$
4 **do** $\rho \leftarrow$ any valid event for π which is good with respect to σ
5 output ρ
6 $\pi \leftarrow \rho\pi$
7 $d \leftarrow d + 1$
8 output d

The complexity of this algorithm is $O(n^2)$, because the main loop is executed at most n times and consumes at most n steps per iteration. Lemma 7.3.2 guarantees that step 4 is well defined.

7.4 Conclusions

We have shown how a classical result on permutation groups leads to a polynomial time algorithm for weighted genome rearrangement distance involving fusions (with weight 1), fissions (with weight 1), and transpositions (with weight 2). This is the first complexity result for a rearrangement problem involving transpositions. We hope this result can be extended to more general problems, involving other events, arbitrary weights, and signed genomes.

Capítulo 8

The Genome Rearrangement Distance Problem with Arbitrary Weights *

Zanoni Dias

João Meidanis

Abstract

Recently we have shown that given two multi-chromossomal genomes it is easy to compute the minimum weight series of fusions, fissions, and transpositions needed to transform one genome into the other, when the weight associated to transpositions is twice as large as that associated to fusions and fissions [91]. In this work we present several results on the computation of distance when an arbitrary weight is associated to transpositions. Some variations of the problem are also studied. For instance we present a polynomial time algorithm for the problem of syntenic distance when only the events of fusions and fissions are admitted.

* *Trabalho depositado como Relatório Técnico no Instituto de Computação da Unicamp em março de 2002, sob o número IC-02-01.*

8.1 Introduction

In the last years genomic science displayed astonishing advances. Hundreds of genomes, from organisms ranging from viruses and uni-cellular organisms such as bacteria all the way up to human beings, had their gene content uncovered [114]. And the mind-boggling rhythm does not stop: every week new genomes are announced. In this context a new challenge emerges: how to relate the huge quantity of genetic information available nowadays?

The area of Genome Rearrangements has progressed a good deal trying to answer this question. In this area we compare two genomes taking into account the order of this genes, rather than the gene sequence as is done in classical sequence comparison. Some studies, for instance, the one done by Palmer and Herbon [101], show that the comparison of gene order leads to conclusions quite compatible with the real evolutionary scenario of the species.

The main rearrangement events are reversals, transpositions, block-interchange, translocations, fusions, and fissions. A reversal inverts a contiguous region of a genome, and flips the orientation of the genes, in case the genome has this information. A transposition exchanges two adjacent regions of a chromosome, while a block-interchange exchanges two arbitrary, contiguous regions of a chromosome. A translocation exchanges regions from distinct chromosomes. A fusion joins two chromosomes together and a fission divides a chromosome into two new ones.

The main results obtained in this area recently are the following. Hannenhalli and Pevzner presented the first polynomial-time algorithm for the reversal distance problem when gene orientations are known [64], with faster and faster algorithms closely following [13, 73, 6]. Caprara proved that the reversal distance problem is NP-Hard when no information on the orientation of the genes is given [21], and in this case, the best approximation algorithm, with a performance ratio of 1.5, was proposed by Christie [29].

The transposition distance problem has been studied by Bafna and Pevzner [9]. They presented an approximation algorithm for the problem with performance ratio of 1.5. Alternative approximation algorithms were proposed by Christie [30] and Walter, Meidanis, and Dias [121]. The complexity of the transposition distance problem still unknown.

The block-interchange distance problem was proposed and solved by Christie [28]. The distance problem involving translocation, fusion, and fission was solved by Hannenhalli and Pevzner [62].

In a previous work, we presented the first polynomial-time algorithm for distance problems involving transposition [91]. We exhibited an algorithm that finds a minimum cost series of the events fusion, fission, and transposition, when a transposition costs twice as much as a fusion or fission, needed to transform a genome in another. The idea of having a larger weight for transpositions came from the fact that experiments have shown that transpositions occurs with about half the frequency of reversals in real biological instances [16]. The main result of this

last work is based on properties of permutation group.

In this work we presente several results on the fusion, fission, and transposition distance problem when the weight associated to transpositions is arbitrary. Variants of the problem are also studied, for instance, the syntenic distance problem when just fusions and fission are allowed. In this case we show a polynomial-time algorithm based on partial results on the syntenic distance using fusions, fissions, and translocations developed by Ferreti and colleagues [48], and later by DasGupta and colleagues [38].

In the following sections we define more formally the problem of computing the fusion, fission, and transposition distance, with emphasis on the case where we associate an arbitrary weight to the transposition event. We present also inequalities involving the versions of the distance with weight two and with another weight associated to transpositions. We then show that the problem of computing the fusion, fission, and transposition distance with arbitrary weight is at least as difficult as computing the transposition distance. We show also a variation of our main problem: the syntenic distance problem with just fusions and fissions. Finally, we present conclusions and ideas for future work.

8.2 The Fusion, Fission, and Transposition Problem

Before attacking the problem with an arbitrary weight given to transpositions, let us formally define the original problem tacked by Meidanis and Dias [91].

Definition 8.2.1 *Given two genomes π and σ, denote by $d(\pi, \sigma)$ the weight of a minimum-weight series of fusions, fissions, and transpositions that transform π into σ, when we associate weight 1 to fusions and fissions, and weight 2 to transpositions.*

In their original work, the authors show that $d(\pi, \sigma)$ can be computed in polynomial time, with an $O(n)$ algorithm, where n is the number of genes in the genome π (or in σ).

A natural question is then: what can we say about this problem when an arbitrary weight ω is used instead of 2 for transpositions? The present work answers this question partially.

8.2.1 Using Arbitrary Weights for Transpositions

Here we consider the fusion, fission, and transposition distance problem between two genomes when the weights associated to the mutational events are 1, 1, and ω, respectively. In this case we denote by $d_\omega(\pi, \sigma)$ the distance between the two genomes.

In the sequel we show a few basic facts about the relation between $d(\pi, \sigma)$ and $d_\omega(\pi, \sigma)$.

Lemma 8.2.1 *For $\omega \geq 2$ we have:*

$$d_\omega(\pi, \sigma) = d(\pi, \sigma).$$

Proof: It is easy to see that every transposition can be replaced by a fission-fusion pair. Therefore, every optimal series of events when transpositions have weight $\omega \geq 2$ is composed solely by fusions and fissions. But by results of Meidanis and Dias [91], every optimal series of fusions and fissions that transforms π into σ has weight $d(\pi, \sigma)$. ■

In the sequel we will show some properties of $d_\omega(\pi, \sigma)$.

Lemma 8.2.2 *For $\omega > 0$ we have:*

$$d_\omega(\pi, \sigma) \leq d(\pi, \sigma).$$

Theorem 8.2.1 *For $0 < \omega \leq 2$ we have:*

$$d(\pi, \sigma) \leq \frac{2}{\omega} d_\omega(\pi, \sigma).$$

Proof: Let $\rho_1, \rho_2, \ldots, \rho_k$ be an optimal series of events such that:

$$\rho_k \rho_{k-1} \cdots \rho_1 \pi = \sigma,$$

considering weight ω for transpositions. If $n_\tau(\rho_1, \rho_2, \ldots, \rho_k)$ is the number of transpositions in the optimal series $\rho_1, \rho_2, \ldots, \rho_k$ and $n_{ff}(\rho_1, \rho_2, \ldots, \rho_k)$ is the number of fusions and fissions in this series, we can write:

$$d_\omega(\pi, \sigma) = n_{ff}(\rho_1, \rho_2, \ldots, \rho_k) + \omega n_\tau(\rho_1, \rho_2, \ldots, \rho_k)$$

However, we can also write:

$$d(\pi, \sigma) \leq n_{ff}(\rho_1, \rho_2, \ldots, \rho_k) + 2n_\tau(\rho_1, \rho_2, \ldots, \rho_k), \tag{8.1}$$

because, after all, $\rho_1, \rho_2, \ldots, \rho_k$ is one possible series of events leading from π to σ. Multiplying Equation 8.1 by $\omega/2$ we get, successively,

$$\begin{aligned} \frac{\omega}{2} d(\pi, \sigma) &\leq \frac{\omega}{2} n_{ff}(\rho_1, \rho_2, \ldots, \rho_k) + \omega n_\tau(\rho_1, \rho_2, \ldots, \rho_k) \\ &\leq n_{ff}(\rho_1, \rho_2, \ldots, \rho_k) + \omega n_\tau(\rho_1, \rho_2, \ldots, \rho_k) \\ &= d_\omega(\pi, \sigma), \end{aligned}$$

since $\omega/2 \leq 1$. ■

From Lemma 8.2.2 and Theorem 8.2.1 we have the following result.

Theorem 8.2.2 *For $0 < \omega \leq 2$ we have:*

$$d_\omega(\pi, \sigma) \leq d(\pi, \sigma) \leq \frac{2}{\omega} d_\omega(\pi, \sigma)$$

Therefore, we may conclude that the algorithm for the fusion, fission, and transposition (with weight 2 for transpositions) presented by Meidanis and Dias [91] is an approximation algorithm with performance ratio $\frac{2}{\omega}$ for the problem where the weight of a transposition is ω, with $0 < \omega < 2$.

The next theorem indicates a sufficient condition for a transposition with weight ω, used on an optimal series that transforms π into σ, to be used on an optimal series that transforms π into σ when transpositions have weight 2.

Theorem 8.2.3 *Let π and σ be two genomes, and τ a transposition. If $d_\omega(\pi,\sigma) = d_\omega(\tau\pi,\sigma) + \omega$, $d(\pi,\sigma) < \frac{1}{(2-\omega)}$ and $1 < \omega < 2$, then $d(\pi,\sigma) = d(\tau\pi,\sigma) + 2$.*

Proof: First we notice that the hypothesis $d_\omega(\pi,\sigma) = d_\omega(\tau\pi,\sigma) + \omega$ implies $\omega \leq 2$, and that $d(\pi,\sigma) < \frac{1}{(2-\omega)}$ implies $\omega > 1$.

Assume that $d_\omega(\pi,\sigma) \geq d_\omega(\tau\pi,\sigma) + \omega$. We have then:

$$\begin{aligned} d(\pi,\sigma) &\geq d_\omega(\pi,\sigma) \\ &\geq d_\omega(\tau\pi,\sigma) + \omega \\ &\geq \frac{\omega d(\tau\pi,\sigma)}{2} + \omega \end{aligned}$$

Rewriting,

$$2d(\pi,\sigma) \geq \omega d(\tau\pi,\sigma) + 2\omega$$
$$d(\pi,\sigma) \geq \omega d(\tau\pi,\sigma) - d(\pi,\sigma) + 2\omega$$
$$d(\pi,\sigma) \geq d(\tau\pi,\sigma) + (\omega-1)d(\tau\pi,\sigma) - d(\pi,\sigma) + 2\omega$$

If we could prove prove that

$$(\omega-1)d(\tau\pi,\sigma) - d(\pi,\sigma) + 2\omega > 1 \tag{8.2}$$

the result follow, because $d(\pi,\sigma)$ is an integer and we would have

$$d(\pi,\sigma) > d(\tau\pi,\sigma) + 1 \geq d(\tau\pi,\sigma) + 2.$$

Therefore, our goal is to prove Equation (8.2). Given that

$$d(\tau\pi,\sigma) \geq d(\pi,\sigma) - 2$$

and because $\omega - 1$ is non-negative, it suffices to prove that

$$(\omega-1)(d(\pi,\sigma) - 2) - d(\pi,\sigma) + 2\omega > 1$$

which, by straighforward algebraic manipulation, is equivalent to

$$d(\pi,\sigma) < \frac{1}{(2-\omega)}$$

provided that $\omega \neq 2$.

■

8.3 Relationship Between Evolutionary Distance Problems

In this section we will show a relationship between the distance problem involving fusion, fission, and transpositions when the weight associated to transpositions is part of the input, and the (pure) transposition distance problem. The transposition problem has been studied intensely in the last years [9, 93, 30, 119], but its computational complexity is still unknown.

Theorem 8.3.1 *The distance problem involving fusions, fissions, and transpositions when the weight of transpositions is part of the input is NP-Hard if the transposition distance problem is NP-Hard.*

Proof: We will show that it is possible to polynomially reduce the transposition distance problem to the distance problem involving fusion, fission, and arbitrarity-weighted transposition (DPFFWT).

Given an instance of the transposition distance problem consisting of two genomes π and σ, we will build an instance for the DPFFWT as follows. We use the genomes π and σ as part of the input, and select $\omega = 1/n$, as the weight to given to transpositions, where n is the number of genes in π (or in σ).

We know that given two genomes A and B with one n-gene chromosome each, it is possible to transform one into the other with at most $n - 1$ transpositions [9]. Therefore, we can solve our instance using transpositions alone at a cost $d_\omega(\pi, \sigma) < 1$, and no fusions or fissions will be used.

The series of events obtained for the DPFFWT is also an optimal series for the transposition distance problem, because of the value chosen for ω. The value of the transposition distance can be obtained as follows: $d_\tau(\pi, \sigma) = nd_\omega(\pi, \sigma)$.

Therefore, we may conclude that if the transposition distance problem is NP-Hard, so is the DPFFWT. ■

Two observations are in order with respect to the value of ω. First, notice that any series of events transforming a mono chromosomal genome π into another monochromosomal genome σ using at least a fusion or a fission will have weight at least 2. Hence, we could have chosen $\omega = 2/n$. Second, if the conjecture by Meidanis, Walter e Dias [93] about the transposition diameter is correct, we would have $D_\tau = \lfloor (n-1)/2 \rfloor + 1$, and hence any value for ω such that $0 < \omega < 4/(n+1)$ would suffice.

In the next section we treat the problem where transpositions have zero weight. Notice that in this case any transformation affecting only one chromosome has cost zero. Therefore, we are in fact interested in guaranteeing that the two genomes have the same sets of genes as chromosomes, regardless of the order of these genes in the set. In other words, we are talking about the synteny problem using only the events of fusion and fission.

8.4 The Syntenic Distance Problem

Ferretti and coworkers [48] proposed a distance measure with a high degree of abstraction, where the order of genes in a particular chromosome is unknown or ignored. The genome of a species is then just a collection of gene sets. Each set correspond to a chromosome. In this synteny context a gene may occur several times in a genome. We define two types of operation: fusion and fission. Fusion correspond to set union, and fission to division of a set A into B and C such that $A = B \cup C$. Notice that B and C may have genes in common. Originally the problem was proposed with a third operation, translocation, which exchanges subsets of two chromosomes.

The *syntenic distance* between two genomes in our context is the minimum number of fusions and fissions necessary to transform the genome of a species into the genome of the other species. We denote by $d_{syntenic}(\pi, \sigma)$ the syntenic distance between genomes π and σ.

Observe that, given two genomes, it is always possible to transform one into the other using only fusions and fissions, when both genomes have the same gene set. The justification of this model is as follows: for many organisms the information that specifies the gene order in a chromosome (physical map) is not known, but the distribution of genes in each chromosome is. Even with such incomplete information it is important to have a precise definition of an evolutionary distance based on genomic events, and the syntenic distance, or just synteny, provides this definition.

DasGupta and colleagues [38] studied the synteny problem when fusions, fissions, and translocations are permitted. They have proved that this problem is NP-Hard, and showed an approximation algorithm with a factor 2. They have also proved that the median problem for three genomes using synteny with the three operations mentioned above is NP-Hard, and that in this case it is possible to obtain approximation algorithms with factor $4 + \epsilon$ for any $\epsilon > 0$. Several of the results presented here on synteny are an adaptation of results from DasGupta et. al. [38] for the problem when only fusions and fission are allowed.

Lemma 8.4.1 *Let π and σ be two genomes with the same gene set. Then we have $d_{syntenic}(\pi, \sigma) = d_{syntenic}(\sigma, \pi)$.*

Proof: Given a series of events transforming π into σ it is easy to revert each operation (the reverse of a fusion is a fission and vice-versa) to obtain a series of operations from σ into π. Hence, the minimum series has the same length in both directions. ■

8.4.1 The Compact Representation

Ferretti, Nadeau and Sankoff [48] defined a compact representation for the synteny problem. Given two genomes π and σ it is possible to obtain a compact representation of the problem

with respect to π using the following method. For each chromosome π_i of the genome π create chromosome $\pi'_i = \{i\}$ in π'. For each chromosome σ_j of σ create $\sigma'_j = \bigcup_{x \in \sigma_j} \{y | x \in \pi_y\}$.

For instance, let $\pi = \{\pi_1, \pi_2, \pi_3\}$ and $\sigma = \{\sigma_1, \sigma_2\}$ with:

$$\pi_1 = \{a, b, c\}, \pi_2 = \{b, d, e\}, \pi_3 = \{f, g\}$$

$$\sigma_1 = \{b, c, d, f, g\}, \sigma_2 = \{a, b, e\}$$

The compact representation of the problem with respect to π is:

$$\pi' = \{\{1\}, \{2\}, \{3\}\}$$

$$\sigma' = \{\{1, 2, 3\}, \{1, 2\}\}$$

Analogously we can define the compact representation with respect to σ. For each chromosome π_i of π, create $\pi''_i = \bigcup_{x \in \pi_i} \{y | x \in \sigma_y\}$. For each chromosome σ_j of σ create $\sigma''_j = \{j\}$ in σ''. The problem above becomes:

$$\pi'' = \{\{1, 2\}, \{1, 2\}, \{1\}\}$$

$$\sigma'' = \{\{1\}, \{2\}\}$$

The following results have been proved by DasGupta et. al. [38].

Lemma 8.4.2 *Let π' and σ' be the two genomes that form the compact representation of π and σ with respect to π. There is a $1-1$ mapping between each operation (fusion or fission) used to transform π into σ and each operation used to transform π' into σ', that is, $d_{syntenic}(\pi, \sigma) = d_{syntenic}(\pi', \sigma')$.*

Given two genomes π and σ, the problem involving the compact representation with respect to π and the problem involving the compact representation with respect to σ are called **dual problems**. The following result shows the relationship between dual problems:

Lemma 8.4.3 *Let π and σ be two genomes over the same set of genes. Let π' and σ' be their compact representation with respect to π, and π'', σ'' be their compact representation with respect to σ. Then $d_{syntenic}(\pi', \sigma') = d_{syntenic}(\pi'', \sigma'')$.*

DasGupta and colleagues [38] have shown also an algorithm to construct the compact representation with respect to a given genome.

Lemma 8.4.4 *If π and σ have n and m chromosomes, respectively, and if each chromosome is a subset of $\{1, 2, \ldots, k\}$, then it is possible to construct the compact representation with respect to π (or with respect to σ) in time $O((k + nm)\alpha(k, n + m))$, where $\alpha(x, y)$ is the inverse of Ackerman's function [34].*

The function $\alpha(x, y)$ grows very slowly, and therefore it is reasonable to expect the algorithm to exhibit a $O(k + nm)$ behaviour in practice.

We define the synteny problem using the compact representation as follows:

Definition 8.4.1 *Let π be a genome with k chromosomes with each chromosome being a subset of $\{1, 2, \ldots, n\}$. The synteny problem is to compute the minimum number of fusion and fission, denoted by $d_{syntenic}(\pi)$, needed to transform π into the genome $\{\{1\}, \{2\}, \ldots, \{n\}\}$.*

8.4.2 The Canonical Order

The synteny distance problem has an important characteristic, rarely found in genome rearrangement problems: a canonical order for the mutation events.

Lemma 8.4.5 *For any series of events that transforms π into σ, with u fusions and v fissions, there is a series transforming π into σ, using $u' \leq u$ fusions and $v' \leq v$ fissions, but where all fusions occur before the fissions.*

Proof:

Let $\rho_1, \rho_2, \ldots, \rho_k$ any sequence of events that transforms π into σ. If all fusions occur before the fissions there is nothing to be done. Let us then assume that there is $i < k$ such that i is the largest integer with ρ_i being a fission and ρ_{i+1} being a fusion. We will construct a new series $\rho_1, \rho_2, \ldots, \rho_{i-1}, \rho'_i, \rho'_{i+1}, \rho_{i+2}, \ldots, \rho_k$, with ρ'_i being a fusion and ρ'_{i+1} being a fission, or conclude that ρ_i and ρ_{i+1} can be supressed. Repeating this procedure as many times as needed we obtain the desired series.

Suppose that fission ρ_i transforms chromosome A of genome $\rho_{i-1}\rho_{i-2}\ldots\rho_1\pi$ into A' e A'', with $A = A' \cup A''$. Likewise, suppose that fusion ρ_{i+1} transforms chromosomes B' and B'' of the genome $\rho_i\rho_{i-1}\ldots\rho_1\pi$ into B, with $B = B' \cup B''$. We have three cases:

- If each of the two chromosomes A' and A'' created by fission ρ_i is different from both B' and B'', then we take $\rho'_i = \rho_{i+1}$ and $\rho'_{i+1} = \rho_i$, since the two events are interchangeable.
- If A' and A'' are the same as B' and B'', then ρ_i and ρ_{i+1} are inverses of each other and can be supressed.
- In the remaining case one of the chromosomes A', A'' is equal to one of B', B''. Without loss of generality, suppose $A' = B'$. We have then that the net effect of ρ_i plus ρ_{i+1} is to transform chromosomes A and B'' of genome $\rho_{i-1}\rho_{i-2}\ldots\rho_1\pi$ into chromosomes A'' and B of genome $\rho_{i+1}\rho_i\ldots\rho_1\pi$. This effect can be also obtained by taking as ρ'_i the fusion that transforms A and B'' into $A \cup B''$, and as ρ'_{i+1} the fission that transforms $A \cup B''$ into A'' and B. This last fission is a valid operation because $A \cup B'' = A'' \cup A' \cup B'' = A'' \cup B' \cup B'' = A'' \cup B$.

■

8.4.3 Lower Bound

In this section we will show a lower bound for the synteny problem using a data structure named synteny graph.

Definition 8.4.2 *Given a genome π, the synteny graph $G_{syntenic}(\pi)$ has one vertex for each chromosome of π. Two vertices are adjacent if and only if the corresponding chromosomes have a nonempty intersection.*

Lemma 8.4.6 *Let π be an arbitrary genome with n genes and p the number of connected components of $G_{syntenic}(\pi)$. Then at least $n - p$ fissions are necessary to transform π into ι_n.*

Proof: By definition $G_{syntenic}(\iota_n)$ has n connected components, and therefore any series of events that transforms π into ι_n must increase the number of connected components by $n - p$. We need to determine how the mutation events affect the synteny graph. A fusion merges two vertices into a single one. If the two vertices are in the same connected component, the number of connected components does not change. If the vertices are in distinct components, then the number of connected components will decrease by one. A fission is the opposite of a fusion, and therefore the number of connected components will either remain the same or increase by one. We conclude that at least $n-p$ fissions are necessary to transform $G_{syntenic}(\pi)$ into a graph with n connected components. ■

Lemma 8.4.7 *Consider a genome π with n genes and c chromosomes. Let p be the number of connected components of the graph $G_{syntenic}(\pi)$. Then at least $c - p$ fusions are needed to transform π into ι_n.*

Proof: Let ρ_1, ρ_2, ..., ρ_k be any series of events transforming π into ι_n with u fusions. According to Lemma 8.4.5, it can be chosen so that the first $l \leq u$ events are fusions and the remaining events are fissions.

Then $\pi_l = \rho_l \rho_{l-1} \ldots \rho_1 \pi$ is a genome that can be transformed into ι_n using fissions only, that is, π_l cannot contain chromosomes A and B with $A \cap B \neq \emptyset$. It follows that every connected component of $G_{syntenic}(\pi_l)$ is composed of a single, isolated vertex. Since fusions do not increase the number of components, the number of connected components in $G_{syntenic}(\pi_l)$ is at most p.

A fusion always decreases the number of vertices by one. We start with c vertices and, after all fusions are applied, we end up with at most p vertices (one vertex per component). It follows that $l \geq c - p$, and, therefore, that $u \geq c - p$. ■

Combining Lemmas 8.4.6 and 8.4.7 we can state the following theorem proposing a lower bound for the syntenic distance using the events of fusion and fission.

Theorem 8.4.1 *Consider a genome π with n genes and c chromosomes. Let p be the number of connected components of $G_{syntenic}(\pi)$. Then $d_{syntenic}(\pi) \geq n + c - 2p$.*

8.4.4 The Polynomial-Time Algorithm

In the sequel we exhibit an algorithm that computes the syntenic distance and yields an optimal series of events that transforms a genome π with n genes into ι_n.

```
SYNTENIC DISTANCE()
 1  Input: π, n
 2  n_fusions ← 0
 3  n_fissions ← 0
 4  Determine the connected components C_1, C_2, ..., C_p of G_syntenic(π)
 5  for i ← 1 to p
 6  do While |C_i| > 1
 7     do ρ ← any fusion involving two chromosomes X and Y of C_i
 8        π ← ρπ
 9        n_fusions ← n_fusions + 1
10        Print ρ
11        Remove X and Y and add X ∪ Y to C_i
12  for j ← 1 to p
13  do While C_j has a chromosome X with more then one gene
14     do ρ ← any fission of X into disjoint parts A and B
15        π ← ρπ
16        n_fissions ← n_fissions + 1
17        Print ρ
18        Remove X and add A and B to C_j
19  Output: n_fusions + n_fissions
```

Figure 8.1: An algorithm for syntenic distance.

Theorem 8.4.2 *Consider a genome π with n genes and c chromosomes. Let p be the number of connected components of the graph $G_{syntenic}(\pi)$. Then $d_{syntenic}(\pi) = n + c - 2p$ and it is possible to obtain an optimal series of events transforming π into ι_n in $O(n^2 + nc\alpha(nc, n))$ time.*

Proof: According to previous results, the algorithm in Figure 8.1 produce a series of events that transforms π into ι_n. Let us compute the number of fusions and fissions used. Each fusion decreases the number of chromosomes by one. Initially, π contains c chromosomes and after all fusions are applied we end up with exactly p chromosomes (one for each component). Therefore we used $c - p$ fusions. Regarding the fissions, each one creates a new chromosome. Because at the end of the algorithm we have n chromosomes, the number of fissions is $n - p$, which implies that $d_{syntenic}(\pi) \leq n + c - 2p$. Using Theorem 8.4.1, we conclude that this is in an exact algorithm for the synteny problem.

The most time-consuming step of the algorithm is the one that determines the connected components of the synteny graph (line 4), without explicitly constructing the graph. This step can be implemented with a union-find structure, using union-by-rank and path compression, in time $O(nc\alpha(nc, n))$ [34]. We construct the chromosome lists of each component in time proportional to the sum of the component sizes, that is, in $O(nc)$ time. Each iteration of the loop on lines 5-11 takes $O(n)$ time, and therefore the entire loop takes $O(nc)$ time, since it is executed $O(c)$ times. Likewise, each iteration of the last loop (lines 12-18) takes $O(n)$ and the total time is $O(n^2)$ because it is executed $O(n)$ times. We conclude that the algorithm runs in $O(n^2 + nc\alpha(nc, n))$ time. ■

8.4.5 The Synteny Problem with Indistinguishable Genes

In the sequel we define a variation for the synteny problem. Here we do not know the order of the genes, nor do we have sufficient information to identify which genes are in which chromosomes. All we know is the number of genes in each chromosome. A chromosome will be represented simply by an integer, and a genome π will be a set of integers (with multiplicity), with $|\pi|$ indicating the number of chromosomes.

A fusion acting on two chromosomes with r and s genes, transforms them into a new chromosome with $t = r + s$ genes. A fission acting on a chromosome with t genes transforms it into two new chromosomes with r and s genes ($t = r + s$).

This model, where we do not have qualitative information on the genes, but only their quantity, is compatible with hybridization experiments involving promoters [49]. These experiments are a fast and simple way of obtaining a good idea on the number of genes in a chromosome.

We define syntenic distance problem between two genomes π and σ with indistinguishable genes as the smallest number of mutation events (fusions and fissions) that transform π into σ, and we denote this distance by $\overline{d}_{syntenic}(\pi, \sigma)$.

Lemma 8.4.8 *Given two arbitrary genomes π and σ, we have $\overline{d}_{syntenic}(\pi, \sigma) \geq |\,|\pi| - |\sigma|\,|$.*

Proof: If $|\pi| > |\sigma|$ at least $|\pi| - |\sigma|$ fusions are needed to transform π into σ, since each fusion decreases the number of chromosomes by one. Likewise, if $|\pi| > |\sigma|$ then at least $|\sigma| - |\pi|$ fissions are needed to transform π into σ, since each fission increases the number of chromosomes by one. ■

Lemma 8.4.9 *Given two arbitrary genomes π and σ, we have $\overline{d}_{syntenic}(\pi, \sigma) \leq |\pi| + |\sigma| - 2$.*

Proof: We can easily transform π into σ using the following algorithm: merge all chromosomes of π into one single chromosome. With $|\pi| - 1$ fusions we can accomplish this transformation. Then apply a series of fission so that the chromosomes of σ are created. For this task $|\sigma| - 1$ fissions suffice. In this way it is possible to transform π into σ using $|\pi| + |\sigma| - 2$ events. ■

Problem	**Result**
Fusion, fission, and transposition distance ($\omega = 2$)	$O(n^2)$ [91]
Fusion, fission, and transposition distance ($0 < \omega < 2$)	Factor $\frac{2}{\omega}$ approx. [HERE]
Synteny with fusion, fission, and translocation	NP-Hard + factor 2 approx. [38]
Synteny with fusion and fission (distinguishable genes)	$O(nc\alpha(nc, n))$ [HERE]
Synteny with fusion and fission (indistinguishable genes)	NP-Hard [HERE]

Table 8.1: Problems and result related to the present work.

Theorem 8.4.3 *Given two genomes π and σ, we have:*

$$| |\pi| - |\sigma| | \leq \overline{d}_{syntenic}(\pi, \sigma) \leq |\pi| + |\sigma| - 2.$$

Theorem 8.4.4 *The syntenic distance problem with indistinguishable genes is NP-Hard.*

Proof: We will show that we can reduce the partition problem polynomially to the syntenic distance problem with indistinguishable genes.

The partition problem can be defined as follows: given a set $A = \{a_1, a_2, \ldots, a_n\}$ of positive integers, determine whether there exists a way of partitioning A into two subsets $B = \{b_1, b_2, \ldots, b_l\}$ and $C = \{c_1, c_2, \ldots, c_m\}$ such that $\sum_{i=1}^{l} b_i = \sum_{j=1}^{m} c_j$.

If $\sum_{i=1}^{n} a_i$ is odd the problem becomes trivial since it is impossible to obtain a suitable partition, but if the sum is even the problem is NP-Hard.

Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set such that $\sum_{i=1}^{n} a_i$ is even. We can construct an instance of the syntenic distance problem with indistinguishable genes as follows: take $\pi = A$ and $\sigma = \{\sigma_1, \sigma_2\}$ where $\sigma_1 = \sigma_2 = (\sum_{i=1}^{n} a_i)/2$.

By Theorem 8.4.3, we have $n - 2 \leq \overline{d}_{syntenic}(\pi, \sigma) \leq n$. Observe that $\overline{d}_{syntenic}(\pi, \sigma) \neq n - 1$, because Lemma 8.4.8 says that $n - 2$ fusions are necessary; if the extra event is a fusion, we end up with 1 chromosome; if it is a fission, we end up with 3 chromosomes; but σ has 2 chromosomes.

The final part of the proof is to show that a solution for this instance of the syntenic distance problem with indistinguishable genes corresponds to a solution of the original partition problem. Notice that, in this context, the ability to partition A into two subsets of equal sum is equivalent to being able to transform π into σ using fusions only.

If $\overline{d}_{syntenic}(\pi, \sigma) = n - 2$ then it is possible to partition A as desired, since π can be transformed into σ using fusions only. In contrast, if $\overline{d}_{syntenic}(\pi, \sigma) = n$, it is impossible to find a suitable partition, because a fission was necessary to transform π into σ. ■

8.5 Conclusion

We have shown in this work results about the rearrangement distance using fusion, fissions, and transpositions when an arbitrary weight is associated to transpositions (see Table 8.1). We have proved that the distance problem using fusions, fissions, and transpositions with the transposition weight given as input is at least as hard as the transposition distance problem, which is still open. Finally, we have determined the complexity of two variations on the syntenic distance problem.

Capítulo 9

Sorting by Prefix Transpositions *

Zanoni Dias

João Meidanis

Abstract

A transposition is an operation that exchanges two consecutive, adjacent blocks in a permutation. A prefix transposition is a transposition that moves the first element in the permutation. In this work we present the first results on the problem of sorting permutations with the minimum number of prefix transpositions. This problem is a variation of the transposition distance problem, related to genome rearrangements. We present approximation algorithms with performance ratios of 2 and 3. We conjecture that the maximum prefix transposition distance is $D(n) = n - \lfloor \frac{n}{4} \rfloor$ and present the results of several computational tests that support this. Finally, we propose an algorithm that decides whether a given permutation can be sorted using just the number of transpositions indicated by the breakpoint lower bound.

* *Trabalho apresentado no* **String Processing and Information Retrieval** *(SPIRE'2002), realizado na cidade de Lisboa, Portugal, em setembro de 2002. Uma versão estendida deste trabalho foi aceita para publicação na revista* **Journal of Discrete Algorithms.**

9.1 Introduction

Sequence comparison is one of the most studied problems in computer science. Usually we are interested in finding the minimum number of local operations, such as insertions, deletions, and substitutions that transform a given sequence into another given sequence. This is the edit distance problem, described in many Computational Biology textbooks [106]. Several studies, however, have shown that global operations such as reversals and transpositions (also called rearrangement events) are more appropriate when we wish to compare the genomes of two species [101].

A new research area called Genome Rearrangements appeared in the last years to deal with problems such as, for instance, to find the minimum number of rearrangement events needed to transform one genome into another. In the context of Genome Rearrangements, a genome is represented by an n-tuple of genes (or gene clusters). When there are no repeated genes, this n-tuple is a permutation. We proceed with a brief overview of the literature related to the present work.

The best studied rearrangement event is the reversal. A reversal inverts a block of any size in a genome. Caprara [18] proved that finding the minimum number of reversals needed to transform one genome into another is an NP-Hard problem. Bafna and Pevzner [8] have presented an algorithm with approximation factor 2 for this problem. Later Christie [29] presented the best known algorithm for the problem, with factor $\frac{3}{2}$.

Hannenhalli and Pevzner [64] have studied the reversal distance problem when the orientation of genes is known. In this case they proved that there is a polynomial algorithm for the problem. This algorithm has been refined successively until Kaplan, Shamir and Tarjan [73] presented a quadratic algorithm. Meidanis, Walter e Dias [95] have shown that all the reversal theory developed for linear genomes can be easily adapted to circular genomes.

Another interesting variation of this problem is the so-called prefix reversal problem or pancake problem as it was originally called [45]. In this variation only reversals involving the first consecutive elements of a genome are permitted. Heydari and Sudborough [66] have proved that this problem is NP-Hard. Gates and Papadimitriou [52] and Heydari and Sudborough [67] have studied the diameter of prefix reversals (see further details on diameter problems in Section 9.4).

The rearrangement event called transposition has the property of exchanging two adjacent blocks of any size in a genome. The transposition distance problem, that is, the problem of finding the minimum number of transpositions necessary to transform one genome into another, has been studied by Bafna and Pevzner [9], who presented the best approximation algorithm for the problem, with factor $\frac{3}{2}$. The transposition distance problem is still open: we do not know of any NP-Hardness proof, and there are no evidences that an exact polynomial algorithm exists. Christie [30] and Meidanis, Walter and Dias [93] have proved partial results on the transposition

diameter.

In this work we present the first known results on the variation of the transposition distance problem that we call prefix transposition distance, that is, the rearrangement distance problem where only transpositions affecting two consecutive blocks of the genome, with one of these blocks formed by the first consecutive elements of the genome.

The paper is divided as follows. Initially, in Section 9.2, we define important concepts that will be used throughout. In Section 9.3 we present two approximation algorithms for the prefix transposition distance problem, with factors 3 and 2. In Section 9.4 we present several results on the prefix transposition diameter, leading to the conjecture that $D(n) = n - \left\lfloor \frac{n}{4} \right\rfloor$, and tests with programs that we implemented to help validate our conjectures. We show in Section 9.5 an algorithm that verifies whether a given genome can be sorted using the minimum number of prefix transpositions according to the breakpoint lower bound (Lemma 9.3.5). Finally, in Section 9.6, we exhibit our conclusions and suggestions for future work.

9.2 Definitions

Here we introduce a number of basic concepts used in Genome Rearrangements. Notice that some definitions, for instance that of transposition, is different from the definition used in other areas.

Definition 9.2.1 *An arbitrary genome formed by n genes will be represented as a permutation $\pi = [\pi[1]\ \pi[2]\ \ldots\ \pi[n]]$ where each element of π represents a gene. The identity genome ι_n is defined as $\iota_n = [1\ 2\ \ldots\ n]$.*

Definition 9.2.2 *A transposition $\rho(x, y, z)$, where $1 \leq x < y < z \leq n+1$, is an rearrangement event that transforms π into the genome $\rho\pi = [\pi[1]\ \ldots\ \pi[x-1]\ \pi[y]\ \ldots\ \pi[z-1]\ \pi[x]\ \ldots\ \pi[y-1]\ \pi[z]\ \ldots\ \pi[n]]$.*

Definition 9.2.3 *A prefix transposition $\rho(1, x, y)$, where $1 < x < y \leq n+1$, is an rearrangement event that transforms π into the genome $\rho\pi = [\pi[x]\ \ldots\ \pi[y-1]\ \pi[1]\ \ldots\ \pi[x-1]\ \pi[y]\ \ldots\ \pi[n]]$.*

Definition 9.2.4 *Given two genomes π and σ we define the transposition distance $d_\tau(\pi, \sigma)$ between these two genomes as being the least number of transpositions needed to transform π into σ, that is, the smallest r such that there are transpositions $\rho_1, \rho_2, \ldots \rho_r$ with $\rho_r \ldots \rho_2\rho_1\pi = \sigma$. We call sorting distance by transpositions, $d_\tau(\pi)$, the transposition distance between the genomes π and ι_n, that is, $d_\tau(\pi) = d_\tau(\pi, \iota_n)$.*

Definition 9.2.5 *Given two genomes π and σ we define the prefix transposition distance $d(\pi, \sigma)$ between these two genomes as being the least number of prefix transpositions needed to transform π into σ, that is, the smallest r such that there are prefix transpositions $\rho_1, \rho_2, \ldots \rho_r$ with $\rho_r \ldots \rho_2\rho_1\pi = \sigma$. We call sorting distance by prefix transpositions, $d(\pi)$, the prefix transposition distance between genomes π and ι_n, that is, $d(\pi) = d(\pi, \iota_n)$.*

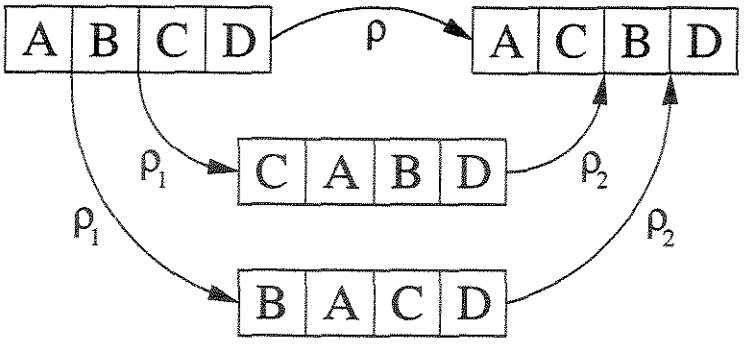

Figure 9.1: Two examples of how it is possible to obtain prefix transpositions ρ_1 and ρ_2 such that $\rho\pi = \rho_2\rho_1\pi$, for a given transposition $\rho = \rho(x, y, z)$, with $x \neq 1$.

9.3 Approximation Algorithms

The first important observation is the following.

Lemma 9.3.1 *For any permutation π, we have $d(\pi) \geq d_\tau(\pi)$.*

Proof: This follows from the observation that every prefix transposition is a transposition. The converse is not always true. ■

9.3.1 Approximation Algorithm with Factor 3

Lemma 9.3.2 *For every transposition $\rho(x, y, z)$ with $x \neq 1$, there are prefix transpositions $\rho_1(1, r, s)$ and $\rho_2(1, t, u)$ such that $\rho_2\rho_1\pi = \rho\pi$.*

Proof: Indeed, it suffices to take $r = y$, $s = z$, $t = z-y+1$ and $u = z-y+x$, or, alternatively, $r = x$, $s = y$, $t = y - x + 1$ and $u = z$. Figure 9.1 shows how two prefix transpositions can simulate a transposition. ■

Lemma 9.3.3 *Any k-approximation algorithm for the transposition distance problem can be transformed into a $2k$-approximation algorithm for the prefix transposition distance problem.*

Proof: Immediate from Lemma 9.3.2. ■

Therefore it is easy to obtain an approximation algorithm with factor 3 for the prefix transposition distance problem using the approximation algorithms with factor $\frac{3}{2}$ for the transposition distance problem given by Bafna and Pevzner [9] and by Christie [30].

9.3.2 Approximation Algorithm with Factor 2

We need to define a few important concepts before proceeding.

Definition 9.3.1 *A breakpoint for the prefix transposition problem is a position i of a permutation π such that $\pi[i] - \pi[i-1] \neq 1$, and $2 \leq i \leq n$. By definition, position 1 (beginning of the permutation) is always considered a breakpoint. Position $n+1$ (end of the permutation) is considered a breakpoint when $\pi[n] \neq n$. We denote by $b(\pi)$ the number of breakpoints of permutation π.*

By the former definition $b(\pi) \geq 1$ for any permutation π and the only permutations with exactly one breakpoint are the identity permutations ($\pi = \iota_n$, for all n).

Definition 9.3.2 *A strip is a subsequence $\pi[i..j]$ of π ($i \leq j$) such that i and $j+1$ are breakpoints and there are no breakpoints between these positions.*

Definition 9.3.3 *Given a permutation π and a prefix transposition ρ, we define $\Delta b(\pi, \rho)$ as the variation on the number of breakpoints due to operation ρ, that is, $\Delta b(\pi, \rho) = b(\rho\pi) - b(\pi)$.*

The first important observation about breakpoints is the following.

Lemma 9.3.4 *Given a permutation π and a prefix transposition ρ, we have that $\Delta b(\pi, \rho) \in \{-2, -1, 0, 1, 2\}$.*

Lemma 9.3.5 *For every permutation π, we have that $d(\pi) \geq \left\lceil \frac{b(\pi)-1}{2} \right\rceil$.*

Proof: Immediate from Lemma 9.3.4. ■

Lemma 9.3.6 *Given a permutation $\pi \neq \iota_n$, where $n = |\pi|$, it is always possible to obtain a prefix transposition ρ such that $\Delta b(\pi, \rho) \geq -1$.*

Proof: Let k be the last element of the first strip of π. If $k < n$, then there is a strip beginning with the element $k+1$, such that $\pi^{-1}[k] < \pi^{-1}[k+1]$ and $\rho = \rho(1, \pi^{-1}[k]+1, \pi^{-1}[k+1])$ suffices. If $k = n$, take $\rho = \rho(1, \pi^{-1}[k]+1, n+1)$. ■

Lemma 9.3.7 *Let π be a permutation and $\rho(1, x, y)$ a prefix transposition such that $\rho\pi = \iota_n$, where $n = |\pi|$. Then $\pi[x] = 1$ and $\Delta b(\pi, \rho) = -2$.*

Lemma 9.3.8 *For every permutation π, we have $d(\pi) \leq b(\pi) - 2$.*

Proof: Immediate by Lemmas 9.3.6 and 9.3.7. ■

Theorem 9.3.1 *For every permutation π, we have $\left\lceil \frac{b(\pi)-1}{2} \right\rceil \leq d(\pi) \leq b(\pi) - 2$.*

Theorem 9.3.2 *Any algorithm that produces the prefix transpositions according to Lemmas 9.3.6 and 9.3.7 is an approximation algorithm with factor 2 for the prefix transposition distance problem.*

Another important point regarding genome rearrangments is the possibility of sorting a permutation without ever increasing the number of breakpoints. Christie [30] has proved that this is true for transposition events. The following lemma establishes the analogous result for prefix transpositions. The proof is a bit lengthy and is omitted here, but appears in the full version of this paper.

Lemma 9.3.9 *Let π be an arbitrary permutation and $d(\pi) = k$ its prefix transposition distance. Then there exists an optimal sequence of prefix transpositions $\rho_1, \ldots, \rho_k$, such that $\rho_k \ldots \rho_1 \pi = \iota_n$, where $n = |\pi|$, and $\Delta b(\rho_{i-1} \ldots \rho_1 \pi, \rho_i) \geq 0$ for every $1 \leq i \leq k$.*

9.3.3 The Cycle Diagram

Bafna and Pevzner [9] developed the "Cycle Diagram" (originally called "Cycle Graph"), a very useful tool in the study of transposition distance problems. This structure was defined also as the "Reality and Desire Diagram" by Meidanis, Walter and Dias [92], a definition that we reproduce below.

Definition 9.3.4 *The vertex sequence of the Cycle Diagram is constructed as follows: for every element $\pi[i]$ of permutation π create a pair $-\pi[i]$ and $+\pi[i]$, in this order. Add a vertex $+0$ in the beginning of the sequence and a vertex $-(n+1)$ in the end. The edges of the diagram are of two types: the "desire" (gray) edges and the "reality" (black) edges. The reality edges are drawn joining vertices $+0$ and $-\pi[1]$, $+\pi[i]$ and $-\pi[i+1]$ (for $1 \leq i \leq (n-1)$), and finally $+\pi[n]$ and $-(n+1)$. The desire edges are drawn joining vertices $+i$ and $-(i+1)$ (for $0 \leq i \leq n$).*

Once defined the diagram, we now need to define its cycles.

Definition 9.3.5 *The size of a cycle in the Cycle Diagram is defined as the number of reality edges that compose the cycle. We call $c(\pi)$ the number of cycles in the Cycle Diagram of π. Similarly, $c_{odd}(\pi)$ is the number of odd cycles in the Cycle Diagram of π.*

Figure 9.2 shows a complete example of a Cycle Diagram. The diagram is composed by two cycles of size 2 and a cycle of size 3. Note that the only permutations of n genes with $n+1$ cycles are the identity permutations ($\pi = \iota_n$, for every n).

Definition 9.3.6 *Given an arbitrary permutation π and a transposition ρ, we define $\Delta c(\pi, \rho)$ ($\Delta c_{odd}(\pi, \rho)$) as the variation in number of cycles (odd cycles) caused by operation ρ, that is, $\Delta c(\pi, \rho) = c(\rho\pi) - c(\pi)$ ($\Delta c_{odd}(\pi, \rho) = c_{odd}(\rho\pi) - c_{odd}(\pi)$).*

The results in the next section were proved by Bafna and Pevzner [9].

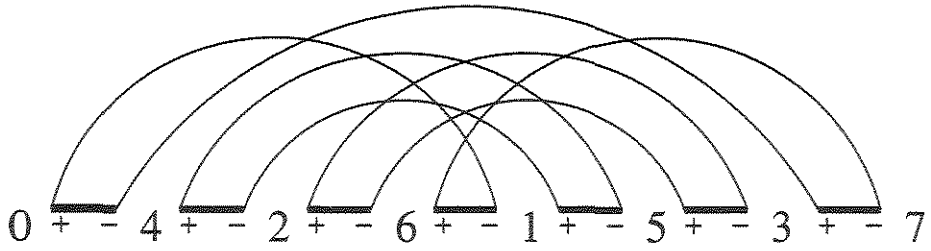

Figure 9.2: Cycle Diagram for $\pi = [4\ 2\ 6\ 1\ 5\ 3]$.

Results for transpositions

Lemma 9.3.10 *Given an arbitrary permutation π and a transposition ρ, we have $\Delta c_{odd}(\pi, \rho) = \{-2, 0, 2\}$.*

Lemma 9.3.11 *For every permutation π, we have $d_\tau(\pi) \geq \frac{n - c_{odd}(\pi)}{2}$.*

Lemma 9.3.12 *Given an arbitrary permutation $\pi \neq \iota_n$, where $n = |\pi|$, it is always possible to obtain either a transposition ρ such that $\Delta c_{odd}(\pi, \rho) = 2$ or three transpositions ρ_1, ρ_2 and ρ_3 such that $\Delta c_{odd}(\pi, \rho_1) = 0$, $\Delta c_{odd}(\rho_1\pi, \rho_2) = 2$ and $\Delta c_{odd}(\rho_2\rho_1\pi, \rho_3) = 2$.*

Lemma 9.3.13 *For every permutation π, we have $d_\tau(\pi) \leq \frac{3}{4}(n - c_{odd}(\pi))$.*

Proof: Immediate by Lemma 9.3.12. ■

Theorem 9.3.3 *For every permutation π, we have $\frac{n - c_{odd}(\pi)}{2} \leq d_\tau(\pi) \leq \frac{3}{4}(n - c_{odd}(\pi))$.*

Theorem 9.3.4 *Any algorithm that produces the transpositions indicated by Lemma 9.3.12 is an algorithm of approximation with factor $\frac{3}{2}$ for the transposition distance problem.*

Results for prefix transpositions

In this section we will prove that it is not possible to obtain an approximation algorithm with factor better than 2 using the theory developed for general transpositions.

Definition 9.3.7 *Let M_k be the family of permutations defined as follows: $M_k = [1\ 3\ 2\ 4\ 6\ 5\ \ldots\ 3k-2\ 3k\ 3k-1]$. The permutation M_k is formed by one cycle of size 1 and k cycles of size 3.*

Lemma 9.3.14 *Consider the permutation M_k, for some $k \geq 1$. In this case we have $d(M_k) \geq 2k$ and it is not possible to obtain a prefix transposition ρ such that $\Delta c_{odd}(M_k, \rho) = 2$ nor three prefix transpositions ρ_1, ρ_2 and ρ_3 such that $\Delta c_{odd}(M_k, \rho_1) = 0$, $\Delta c_{odd}(\rho_1 M_k, \rho_2) = 2$ and $\Delta c_{odd}(\rho_2\rho_1 M_k, \rho_3) = 2$.*

ALGORITHM TO SORT $M_k()$
1 Input: $\pi = M_k$, with $k \geq 1$
2 **for** $i \leftarrow 1$ to k
3 **do** $\pi \leftarrow \rho(1, 3(i-1)+2, 3(i-1)+3)\pi$
4 $\pi \leftarrow \rho(1, 2, 3(i-1)+4)\pi$
5 Output: $2k$

Figure 9.3: Algorithm to sort M_k.

Proof: Each one of the cycles of size 3 is eliminated only when a transposition acts on its three reality edges. Since initially no cycle intercepts any other cycle and no 3-cycle can be immediately destroyed, at least two movements are necessary per cycle. Hence $d(M_k) \geq 2k$. ■

Lemma 9.3.15 *Given a permutation M_k, with $k \geq 1$, we have $d(M_k) \leq 2k$.*

Proof: Immediate by the algorithm of Figure 9.3. A step-by-step execution of this algorithm on permutation M_2 can be seen in Figure 9.4. ■

Theorem 9.3.5 *Given a permutation M_k, for some $k \geq 1$, we have $d(M_k) = 2k$.*

Proof: Immediate by Lemmas 9.3.14 and 9.3.15. ■

Theorem 9.3.6 *No approximation algorithm for the prefix transposition distance problem based on the Cycle Diagram and using the lower bound of Lemma 9.3.11 can have an approximation factor less than 2.*

Proof: It suffices to notice that the lower bound of Lemma 9.3.11 gives $d(M_k) \geq k$, while we know that in fact $d(M_k) = 2k$ by Theorem 9.3.5. ■

9.4 The Diameter of Prefix Transpositions

We call rearrangement diameter the largest rearrangement distance between two permutations of a certain size n. We Denote by $D(n)$ the diameter of prefix transpositions and by $D_\tau(n)$ the diameter of transpositions. Bafna and Pevzner [9] proved the following result.

Theorem 9.4.1 *The diameter of transpositions for permutations of size n is such that $\frac{n}{2} \leq D_\tau(n) \leq \frac{3n}{4}$.*

We can present a similar result for the prefix transposition distance problem.

Theorem 9.4.2 *The diameter of prefix transpositions for permutations of size n is such that $\frac{n}{2} \leq D(n) \leq n-1$.*

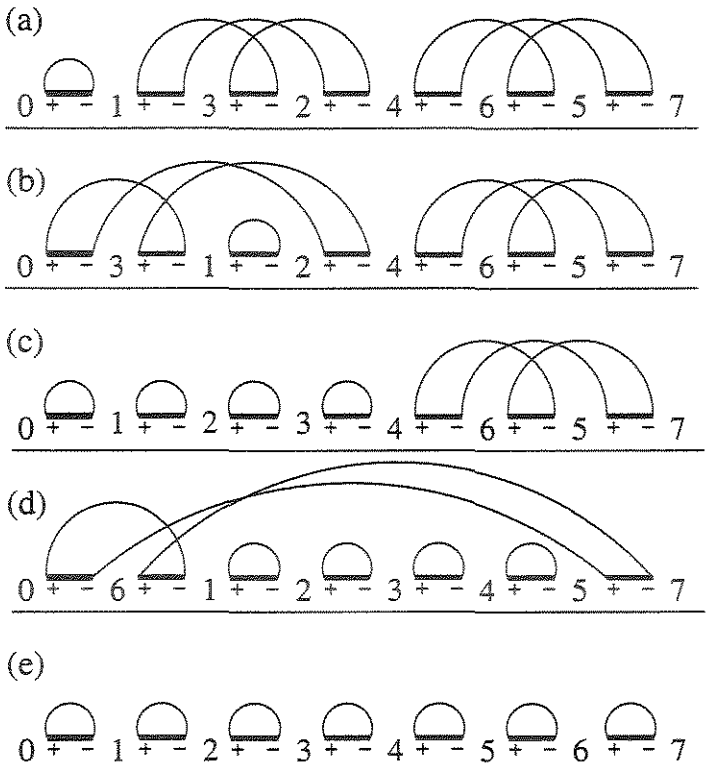

Figure 9.4: (a) Permutation $M_2 = [1\ 3\ 2\ 4\ 6\ 5]$. (b)-(e) Sorting M_2.

Proof: To begin with note that $D(n) \geq D_\tau(n)$, since $d(\pi) \geq d_\tau(\pi)$ for any permutation π (Lemma 9.3.1). We can then use the result of Aigner and West [3] that says that the diameter for the rearrangement distance problem that considers only insertion of the first element, that is, transpositions of the form $\rho(1, 2, x)$, is $n - 1$. ■

Definition 9.4.1 *Let R_n be the family of permutations defined as follows: $R_n = [n\ n{-}1\ \ldots\ 2\ 1]$. Permutation R_n is formed by odd cycles only: just one cycle when n is odd, and two cycles otherwise.*

The following result was proved independently by Christie [30] and Meidanis, Walter and Dias [93].

Theorem 9.4.3 *For $n \geq 3$, we have $d_\tau(R_n) = \left\lfloor \frac{n}{2} \right\rfloor + 1$.*

When dealing with prefix transpositions, we could state, based solely on Theorem 9.3.1, that $\left\lceil \frac{n}{2} \right\rceil \leq d(R_n) \leq n - 1$. However, a stronger statement holds.

Theorem 9.4.4 *For $n \geq 4$, we have $d(R_n) \leq n - \lfloor \frac{n}{4} \rfloor$.*

ALGORITHM TO SORT $R_n()$
1 Input: $\pi = R_n$, with $n \geq 4$
2 $m \leftarrow 4\lfloor \frac{n}{4} \rfloor$
3 {Phase 1: Shuffling}
4 **for** $i \leftarrow 1$ to $(\frac{m}{4}) - 1$
5 **do** $\pi \leftarrow \rho(1, 5, m - 2(i-1))\pi$
6 $\pi \leftarrow \rho(1, 3, \frac{m}{2} + 2)\pi$
7 {Phase 2: Greedy Phase}
8 $x \leftarrow \pi^{-1}[n] + 1$
9 $y \leftarrow m + 1$
10 **for** $i \leftarrow 1$ to $2(\frac{m}{4})$
11 **do** $z \leftarrow \pi[x]$
12 $\pi \leftarrow \rho(1, x, y)\pi$
13 $y \leftarrow \pi^{-1}[z-1] + 1$
14 $w \leftarrow \pi[y] - 1$
15 $x \leftarrow \pi^{-1}[w] + 1$
16 {Phase 3: Positioning the Last Elements}
17 **for** $i \leftarrow (m+1)$ to n
18 **do** $\pi \leftarrow \rho(1, i, i+1)\pi$
19 Output: $n - \lfloor \frac{n}{4} \rfloor$

Figure 9.5: Algorithm to sort R_n.

Proof: The algorithm of Figure 9.5 sorts R_n using exactly $n - \left\lfloor \frac{n}{4} \right\rfloor$ prefix transpositions. A step-by-step execution of this algorithm on permutation R_{13} can be seen in Figure 9.6. ■

Lemma 9.4.1 *For $n \geq 1$, we have $d(R_{n+1}) \geq d(R_n)$.*

Proof: It is easy to see that any series of prefix transpositions that sorts R_{n+1} will also sort R_n, provided we adapt the movements that include the element $n+1$. ■

Christie [30] and Meidanis, Walter and Dias [93] have proposed the following conjecture, still open today.

Conjecture 9.4.1 *The transposition diameter $D_\tau(n)$, for $n \geq 3$, is given by $D_\tau(n) = d_\tau(R_n) = \left\lfloor \frac{n}{2} \right\rfloor + 1$.*

Likewise, we believe that the following statement is true.

Conjecture 9.4.2 *The diameter of prefix transpositions $D(n)$, for $n \geq 4$, is given by $D(n) = d(R_n) = n - \left\lfloor \frac{n}{4} \right\rfloor$.*

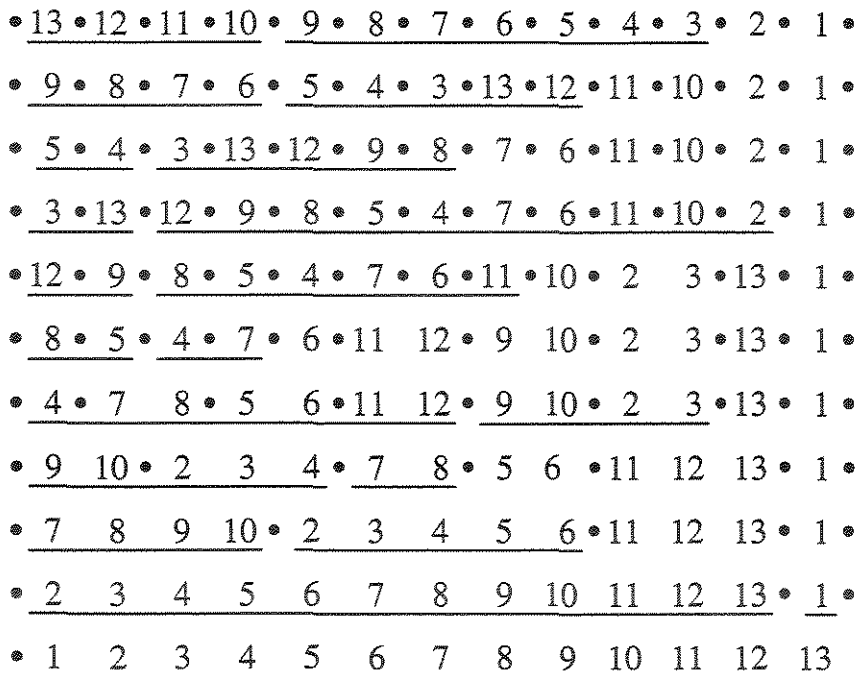

Figure 9.6: Steps to sort R_{13}.

9.4.1 Tests

The tests that will be presented in this section were performed in a Digital Alpha Server GS140 computer, with 10 Alpha 21264 EV6 processors of 524MHz and 64-bit word length, with 8 GB of physical memory and running the OSF1 version 4.0 operating system. All programs were written in C++ and compiled with g++ using compilation directive "-O3". Our programs use just one processor and during the tests the machine was always executing other processes as well. The measured times are the times effectively spent by the programs (user + system time) and not the total time of execution (real time).

We implemented two "branch and bound" algorithms to compute the exact distance of prefix transpositions. The first version considers all possible prefix transpositions, while the second version considers only prefix transpositions that do not create new breakpoints, according to Lemma 9.3.9. Using these programs it was possible to obtain directly the prefix transposition distance for all reverse permutations R_n with $n \leq 15$. Table 9.1 and Figure 9.7 show result summaries.

To verify the correctness of Theorem 9.4.4, we implemented the algorithm that sorts reverse permutations R_n in polynomial time (Figure 9.5). We tested our implementation using all reverse permutations R_n for $n \leq 50000$. The algorithm correctly sorted all tested instances. Note that these instances are several times bigger than the biggest instances used in practice in genome rearrangement problems. Execution times for this algorithm are plotted in Figure 9.8.

n	$d(R_n)$	Time without optimization (seconds)	Time with optimization (seconds)
02	01	0	0
03	02	0	0
04	03	0	0
05	04	0	0
06	05	0	0
07	06	0	0
08	06	4	2
09	07	9	3
10	08	59	22
11	09	1011	373
12	09	8872	2607
13	10	16294	4305
14	11	118463	45168
15	12	2771374	1081631
16	12*	750 days *	300 days *

Table 9.1: distance of prefix transposition for reverse permutations with 16 or less elements. The times in column "without optimization" refer to the "branch and bound" algorithm that considers all prefix transpositions possible, while the column "with optimization" presents the results of the implementation that considers only prefix transpositions that do not create new breakpoints, according to Lemma 9.3.9. We could not compute $d(R_{16})$ directly using any of the two implementations; instead we present an estimate of the time necessary for each algorithm to compute correctly the distance. Note also that it is possible to infer the distance $d(R_{16})$ from Theorem 9.4.4 and Lemma 9.4.1.

Lastly we implemented two programs to verify the conjectures proposed in Section 9.4. The two programs are based in the same strategy. We built a graph as follows: we created a vertex for each of the $n!$ permutations with n elements and an edge for each pair of permutations that differ by a rearrangement event. In this graph we search for the permutations that posses the largest distance from the identity permutation. This strategy can be implemented in linear time on the graph size. With this method we could certify in slightly over 20 hours that both conjectures are true for permutations with $n \leq 11$ elements. Unfortunately 30 GB of physical memory are need to build the graph for $n = 12$, what made the test of our conjectures for $n \geq 12$ impossible.

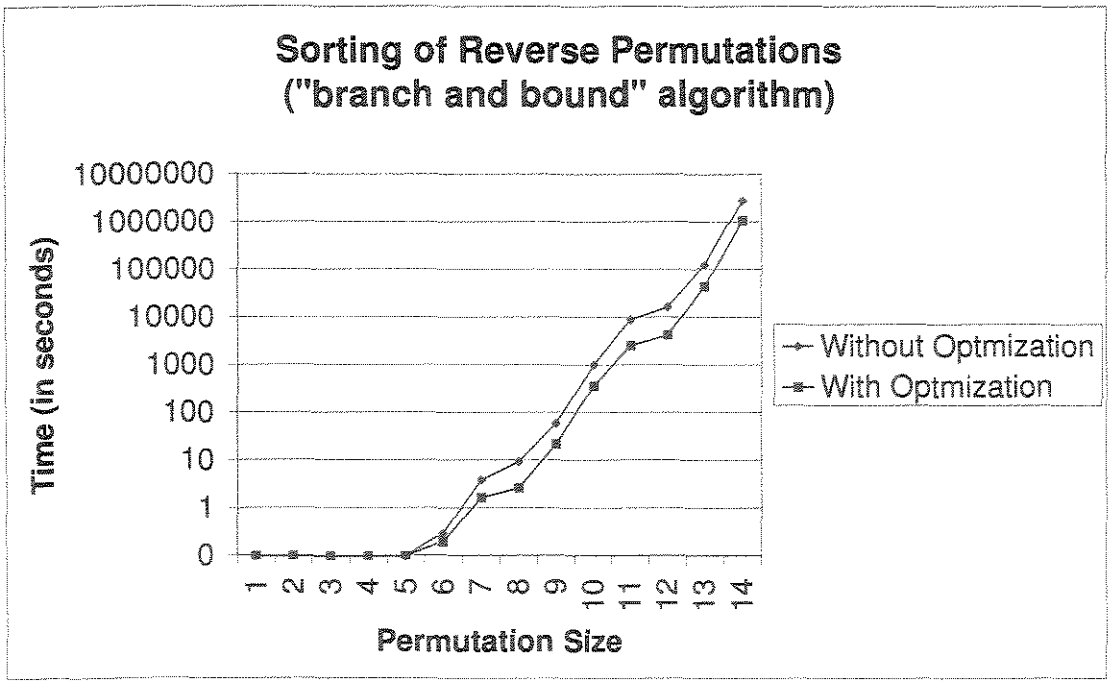

Figure 9.7: Results for the "branch and bound" algorithm. Total approximate time of 47 days nonstop processing, with about 34 days for the version not optimized and 13 days for the optimized version.

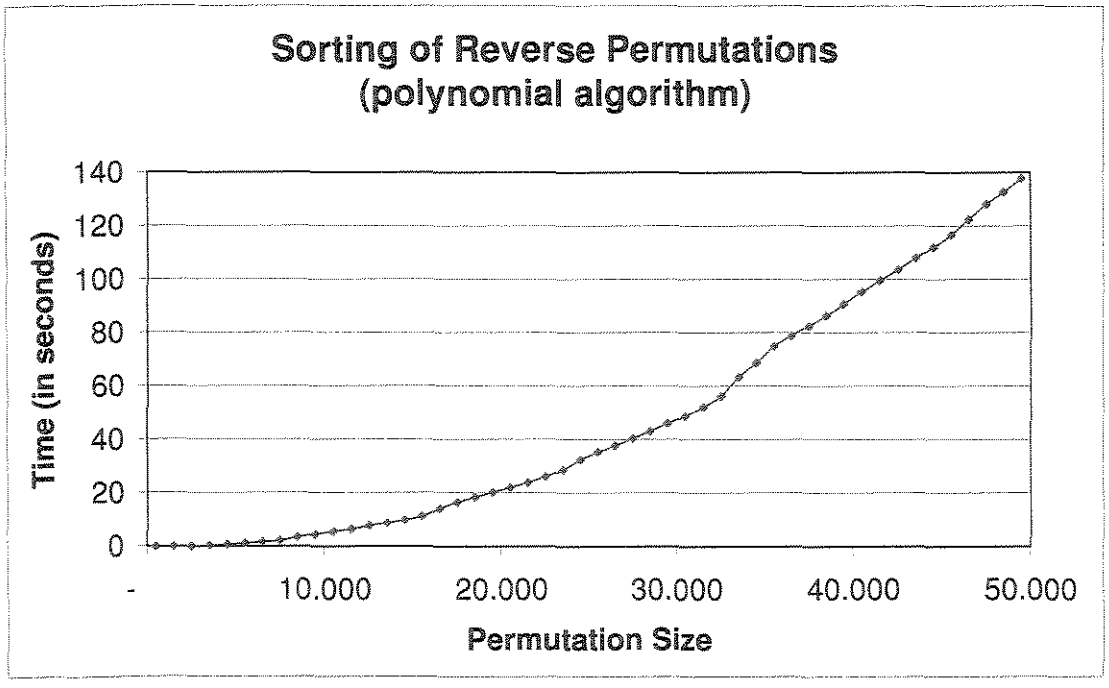

Figure 9.8: Results for the polynomial algorithm. Total approximate time of 27 days nonstop processing.

9.5 Permutations that Satisfy the Breakpoint Lower-Bound

Kececioglu and Sankoff [78] conjectured that to determine whether a permutation can be sorted using the minimum number of reversals indicated by the breakpoint lower bound for reversals was an NP-Hard problem, just like the general problem of sorting by reversals. Irving and Christie [69] and Tran [115] independently proved that this conjecture is false, exhibiting a polynomial algorithm for the problem.

In the case of prefix transpositions we know from Lemma 9.3.5 that for every permutation π we have $d(\pi) \geq \lceil (b(\pi) - 1)/2 \rceil$. However, given a permutation π, is it possible to determine whether $d(\pi) = (b(\pi) - 1)/2$? The following results prove that the answer is yes.

Lemma 9.5.1 *Let π be an arbitrary permutation. Then there exists at most one prefix transposition ρ such that $\Delta b(\pi, \rho) = -2$.*

Proof: Suppose that π and $\rho(1, x, y)$ are such that $\Delta b(\pi, \rho) = -2$. In this case we have $\pi = [\pi[1] \ldots \pi[x-1]\pi[x] \ldots \pi[y-1]\pi[y] \ldots]$ and $\rho\pi = [\pi[x] \ldots \pi[y-1]\ \pi[1] \ldots \pi[x-1]\pi[y] \ldots]$, where $\pi[x-1] \neq \pi[x] - 1$, $\pi[y-1] \neq \pi[y] - 1$, $\pi[y-1] = \pi[1] - 1$ and $\pi[x-1] = \pi[y] - 1$. Finally, note that $\pi[1]$ determines uniquely the index y, and y determines uniquely the index x. ■

Theorem 9.5.1 *Let π be an arbitrary permutation. Then it is possible to determine in polynomial time whether $d(\pi) = \frac{b(\pi)-1}{2}$.*

Proof: Immediate by the algorithm of Figure 9.9, that has complexity $O(n^2)$. ■

Given an integer k, is it always possible to find a permutation π such that there is a series of k prefix transpositions $\rho_1, \ldots, \rho_k$ with $\Delta b(\rho_{i-1}\rho_{i-2} \ldots \rho_1\pi, \rho_i) = -2$, for $1 \leq i \leq k$? Once again the answer is affirmative.

Definition 9.5.1 *Let B_k be the family of permutations defined as follows: B_k = [$k+1$ k $k+2$ $k-1$ $k+3$ $k-2$ ... $2k-1$ 2 $2k$ 1]. Permutation B_k possesses $2k+1$ breakpoints.*

Lemma 9.5.2 *For every integer k it is possible to obtain a series of k prefix transpositions ρ_1, ρ_2, ..., ρ_k that sort B_k such that $\Delta b(\rho_{i-1}\rho_{i-2} \ldots \rho_1\pi, \rho_i) = -2$, for $1 \leq i \leq k$.*

Proof: Immediate from the algorithm of Figure 9.10, that can be implemented in linear time. ■

VERIFYING WHETHER π SATISFIES THE BREAKPOINTS LOWER-BOUND()
1 Input: π
2 $n \leftarrow |\pi|$
3 OK $\leftarrow$ TRUE
4 **While** $\pi \neq \iota_n$ and OK
5 **do** $y \leftarrow \pi^{-1}[\pi[1]-1]+1$
6 $x \leftarrow \pi^{-1}[\pi[y]-1]+1$
7 {Verifies whether there exists a movement that removes two breakpoints}
8 **if** $x < y$
9 **then** $\pi \leftarrow \rho(1,x,y)\pi$
10 **else** OK $\leftarrow$ FALSE
11 Output: OK

Figure 9.9: The algorithm that verifies whether π has distance $d(\pi) = \frac{b(\pi)-1}{2}$.

ALGORITHM TO SORT B_k()
1 Input: $\pi = B_k$, with $k \geq 1$
2 **for** $i \leftarrow 1$ to k
3 **do** $\pi \leftarrow \rho(1, 2i, 2i+1)\pi$
4 Output: k

Figure 9.10: The algorithm that sorts B_k.

9.6 Conclusions

We introduced in this work a new problem of Genome Rearrangement that we called distance of prefix transpositions. We showed a number of results for this problem, including two approximation algorithms (the best of them with factor 2), a proof that any permutation can be sorted without "cutting strips," a conjecture on the prefix transposition diameter stating that $D(n) = n - \left\lfloor \frac{n}{4} \right\rfloor$, and an algorithm for determining whether a permutation can be sorted using a series of prefix transpositions removing two breakpoints per step. The problem remains open.

Capítulo 10

Conclusão

Apresentamos, nos capítulos anteriores, vários trabalhos relacionados a Rearranjo de Genomas. Agrupamos nossas principais contribuições em dois grupos: um novo formalismo algébrico e uma série de resultados envolvendo o evento de transposição.

Esta tese é, de certa forma, uma continuação natural da tese de doutorado da Profa. Maria Emília M. T. Walter [119]. Sendo assim, nossa primeira contribuição está relacionada com o principal problema apontado nas conclusões daquela tese - a falta de uma "teoria estabelecida e uniforme" para Rearranjo de Genomas. Ainda segundo a Profa. Maria Emília, "nesta área, as estruturas e os algoritmos propostos são completamente independentes uns dos outros, inclusive na notação, e as estratégias de soluções vêm de diferentes áreas como Teoria da Computação, Estatística e Matemática".

No Capítulo 6, relacionamos a teoria de Rearranjo de Genomas com a teoria algébrica de grupos de permutações. Nossa intenção foi estabelecer um formalismo algébrico forte o bastante para permitir simplificar a obtenção de novos resultados, até então muito baseados na construção exaustiva de diagramas.

Um genoma foi definido como sendo uma permutação. As diferenças entre dois genomas π e σ são representadas pela fórmula $\sigma\pi^{-1}$, que nos permite deduzir algebricamente uma série de informações relacionadas a distância de rearranjo entre os dois genomas, por exemplo o número de *breakpoints* e de ciclos.

Definimos algebricamente as operações de reversão, transposição, transposição com reversão e *block interchange*. Usando esta teoria foi possível definir também algumas estruturas importantes para o problema da distância de reversão como, por exemplo, ciclos bons e componentes boas.

Consideramos que demos os primeiros passos em direção a estabelecer um novo formalismo para a área, mas muito trabalho ainda precisa ser feito. Uma abordagem interessante seria obter provas baseadas em argumentos algébricos para todos os principais resultados envolvendo o evento de reversão, que já foi vastamente estudado. Esperamos que, num futuro próximo, todos

os resultados apresentados nesta tese possam ser demonstrados usando-se apenas álgebra.

Outra contribuição importante desta tese são os vários resultados envolvendo o evento de transposição. Este evento foi estudado de várias formas diferentes: como um problema de rearranjo de genomas simples (Capítulo 3); associado com outros eventos de rearranjo (Capítulos 4, 5, 7 e 8); e como evento restrito (Capítulo 9).

No Capítulo 3, mostramos um limite inferior para o valor do diâmetro de transposição. Nos dois capítulos seguintes, estudamos o problema da distância de rearranjo em que os eventos de transposição e reversão são permitidos. No Capítulo 4 apresentamos dois algoritmos de aproximação para este problema, e no Capítulo 5 exibimos um limite inferior para o valor do diâmetro.

Apresentamos os primeiros resultados inéditos no Capítulo 7, no qual utilizamos parte do formalismo algébrico recém desenvolvido para mostrar que é possível determinar a distância de fusão, fissão e transposição em tempo polinomial, quando uma transposição tem peso duas vezes maior que uma fusão e uma fissão. Este é o primeiro resultado polinomial conhecido para um problema de rearranjo envolvendo o evento de transposição.

Em seguida, no Capítulo 8, determinamos aproximações para o valor da distância de fusão, fissão e transposição, quando as fusões e fissões têm peso unitário e as transposições têm pesos arbitrários.

No mesmo capítulo, apresentamos dois resultados importantes sobre distância sintênica envolvendo fusões e fissões. Primeiro, provamos que o problema com genes distinguíveis é polinomial e pode ser resolvido de forma eficiente. Em seguida, demonstramos que a versão do problema com genes indistinguíveis é NP-Difícil.

No Capítulo 9, introduzimos o problema da distância de transposição de prefixos. Entre os vários resultados apresentados neste capítulo, destacamos dois algoritmos de aproximação, um algoritmo para testar se uma permutação pode ser ordenada usando transposições de prefixos que sempre removem dois *breakpoints*, e uma conjectura sobre o valor do diâmetro para este problema, baseado num algoritmo que ordena a permutação reversa.

Apesar de todos os esforços, o problema geral de distância de transposição permanece em aberto, não sendo conhecido, até o momento, nenhum algoritmo polinomial exato, nem uma prova que este problema seja NP-Difícil. Temos esperança que o formalismo algébrico introduzido nesta tese possa ser utilizado para a obtenção de novos resultados para transposições.

Vários problemas interessantes de Rearranjo de Genomas não foram abordados nesta tese. Este é o caso, por exemplo, do problema da mediana, que pode ser formulado da seguinte forma. Dados genomas π_1, π_2, ..., π_n determinar um genoma σ tal que $\sum_{i=1}^{n} d(\pi_i, \sigma)$ seja mínima, onde $d(\pi_i, \sigma)$ é a distância de rearranjo entre π_i e σ. Caprara [19] provou que o problema da mediana, quando utilizamos distância de reversão, é NP-Difícil. Existem poucos resultados para o problema da mediana envolvendo outros eventos de rearranjo. Acreditamos que este problema represente uma promissora área de estudos.

Apêndice A

Implementações e Testes

Paralelamente à pesquisa teórica, implementamos alguns programas que poderão auxiliar novos trabalhos na área de Rearranjo de Genomas, sobretudo com relação ao evento de transposição. Podemos dividir as implementações em dois grupos descritos a seguir.

Logo no começo desta tese, implementamos um visualizador da estrutura denominada "Diagrama de Ciclos", estrutura muito utilizada em problemas de Rearranjo de Genomas. Desenvolvemos este visualizador usando uma interface *Web*, de forma que ele pudesse ser acessado em qualquer lugar do mundo conectado a internet.

Mais tarde, escrevemos uma série de programas para o cálculo da distância de transposição. Implementamos os algoritmos de aproximação propostos nesta tese, como também algumas heurísticas e algoritmos exatos *branch and bound* para o problema da distância de transposição.

A seguir descreveremos em mais detalhes cada um desses programas.

A.1 O Visualizador de Diagramas de Ciclos

Um dos maiores problemas no estudo de Rearranjo de Genomas é o fato de grande parte da teoria existente basear-se na construções de diagramas. Assim, para estudar uma seqüência de operações que transforma um genoma em outro é necessário, muitas vezes, desenhar uma série de diagramas, o que deixa a pesquisa muito lenta ou até mesmo inviável.

Aqui vimos que havia duas opções: ou desenvolver uma nova teoria que não usasse diagramas ou qualquer outro recurso gráfico, ou implementar um visualizador automático que, dada uma permutação, desenhasse o diagrama adequado. Escolhemos atuar nas duas frentes. A teoria está sendo desenvolvida, e um resultado inicial é apresentado no Capítulo 6 desta tese.

Batizamos nosso visualizador de *Permutation Info*. Este programa recebe uma permutação, constrói o seu "Diagrama de Ciclos" (também conhecido na literatura como "Diagrama Realidade Desejo" [93]) e exibe o resultado em uma página *web*.

O diagrama é composto por dois tipos de arestas: as cinzas (conhecidas também como

arestas desejo) e as pretas (as arestas realidade). As arestas cinzas são representadas, no nosso visualizador, por arcos, enquanto as arestas pretas, por traços horizontais. Para maiores detalhes sobre este diagrama, consultar o Capítulo 3 desta tese ou o trabalho de Bafna e Pevzner [9] sobre distância de transposição.

Dada uma permutação π qualquer, nosso visualizador apresenta outras informações muito importantes para a investigação do problema da distância de transposição, como:

- o número de *breakpoints* ($b(\pi)$),
- o número de ciclos ($c(\pi)$),
- o número de ciclos ímpares ($c_{odd}(\pi)$),
- a *lower bound* de ciclos ímpares (ou seja, $d(\pi) \geq LB = \frac{n+1-c_{odd}(\pi)}{2}$),
- a *upper bound* baseada no algoritmo de aproximação $\frac{3}{2}$ proposto por Bafna e Pevzner (ou seja, $d(\pi) \leq UB = \left\lceil \frac{3}{4}(n+1-c_{odd}(\pi)) \right\rceil$).

Além disso o *Permutation Info* permite aplicar uma transposição à permutação. Para isso, basta que o usuário forneça o número de três arestas cinzas para indicar onde a permutação será quebrada e rearranjada pela transposição.

Um exemplo prático do uso do *Permutation Info* pode ser acompanhado pelos "Diagrama de Ciclos" e pelas outras informações exibidas nas figuras A.1 até A.6, onde uma série ótima de transposições é usada para transformar a permutação $\pi = [8\ 7\ 3\ 2\ 1\ 6\ 5\ 4]$ na permutação identidade $\iota_8 = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$.

O *Permutation Info* foi escrito em Perl de forma a permitir que ele pudesse ser utilizado em qualquer sistema computacional com um servidor *web* adequadamente configurado e com um interpretador Perl padrão. Nosso visualizador usa as bibliotecas CGI e GD, ambas de domínio público [35].

A.2 Calculando a Distância de Transposição

A seguir, descreveremos os algoritmos que implementamos para calcular a distância de transposição. Dividimos esta seção em duas partes, na primeira, abordaremos dois algoritmos usados para calcular a distância exata, enquanto na segunda parte, falaremos da implementação de uma heurística para este problema. Todos os algoritmos apresentados nesta seção foram implementados em Perl e posteriormente portados para C++ usando STL [109].

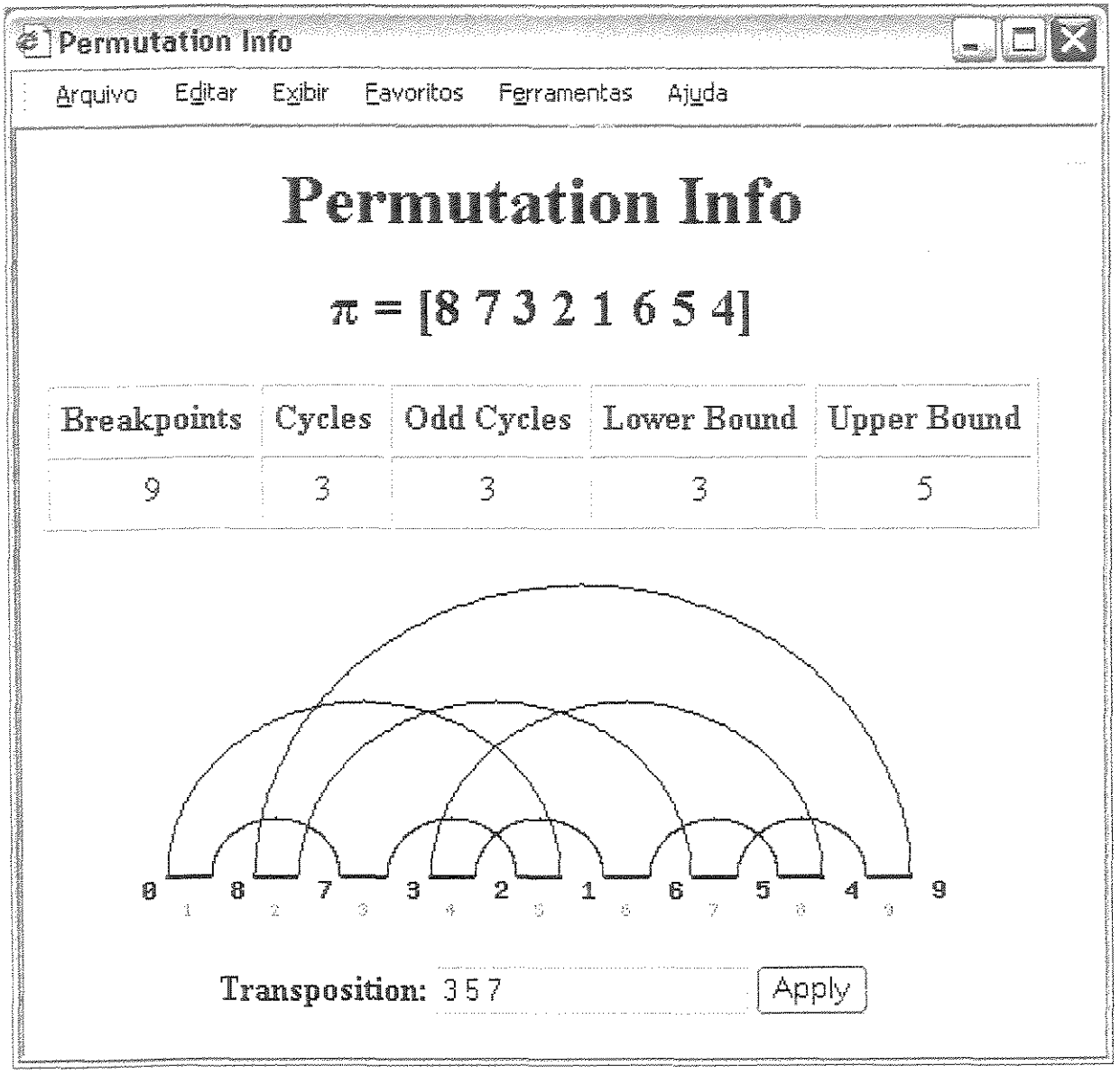

Figura A.1: *Permutation Info*: neste exemplo, vemos o "Diagrama de Ciclos" da permutação π = [8 7 3 2 1 6 5 4], formado por três ciclos ímpares, antes que os blocos [3 2] e [1 6] fossem trocados de lugar pela transposição $\rho(3, 5, 7)$.

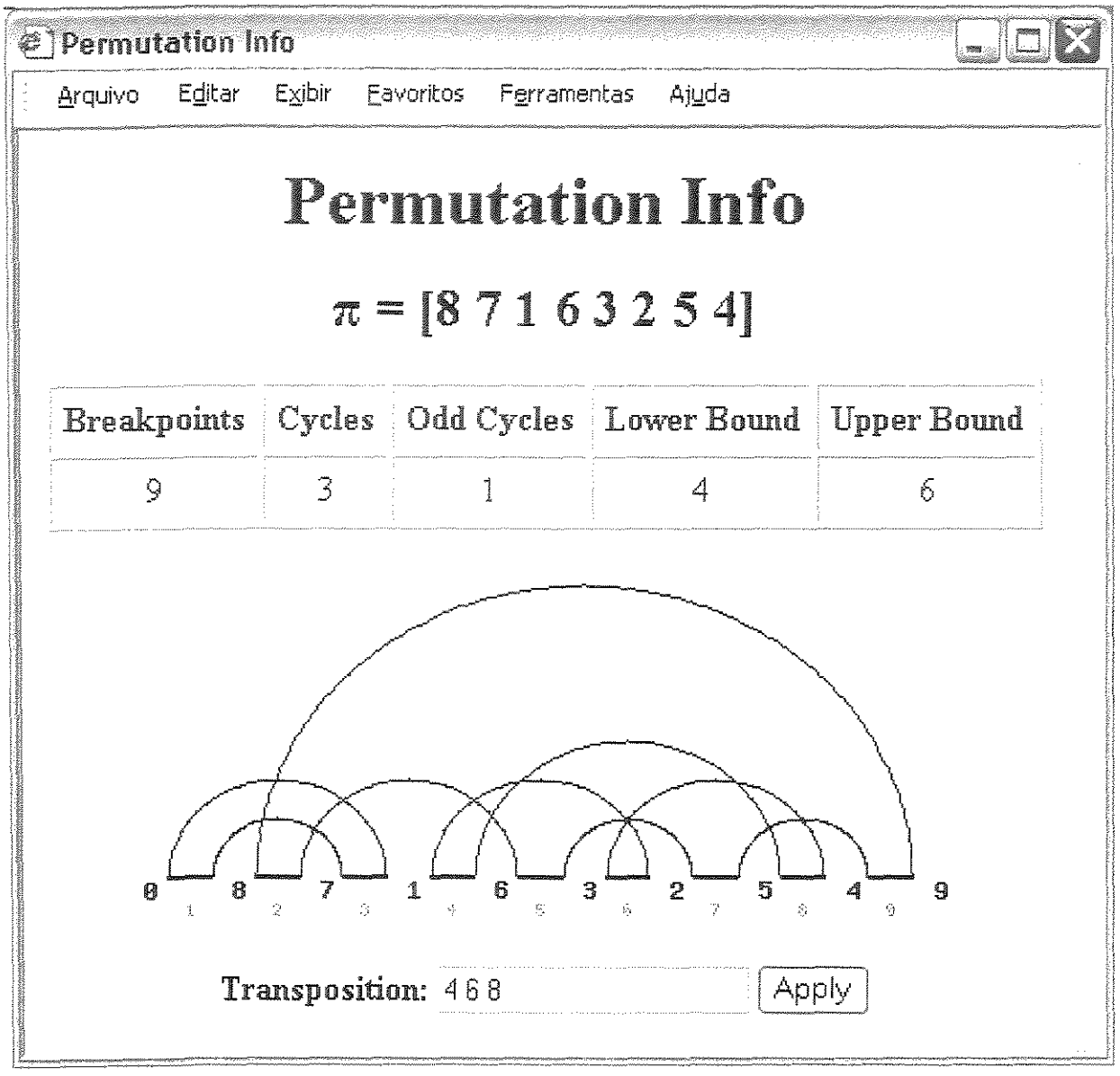

Figura A.2: *Permutation Info*: neste exemplo, vemos o "Diagrama de Ciclos" da permutação π = [8 7 1 6 3 2 5 4], formado por três ciclos, sendo dois deles pares e apenas um ímpar, antes que os blocos [6 3] e [2 5] fossem trocados de lugar pela transposição $\rho(4, 6, 8)$.

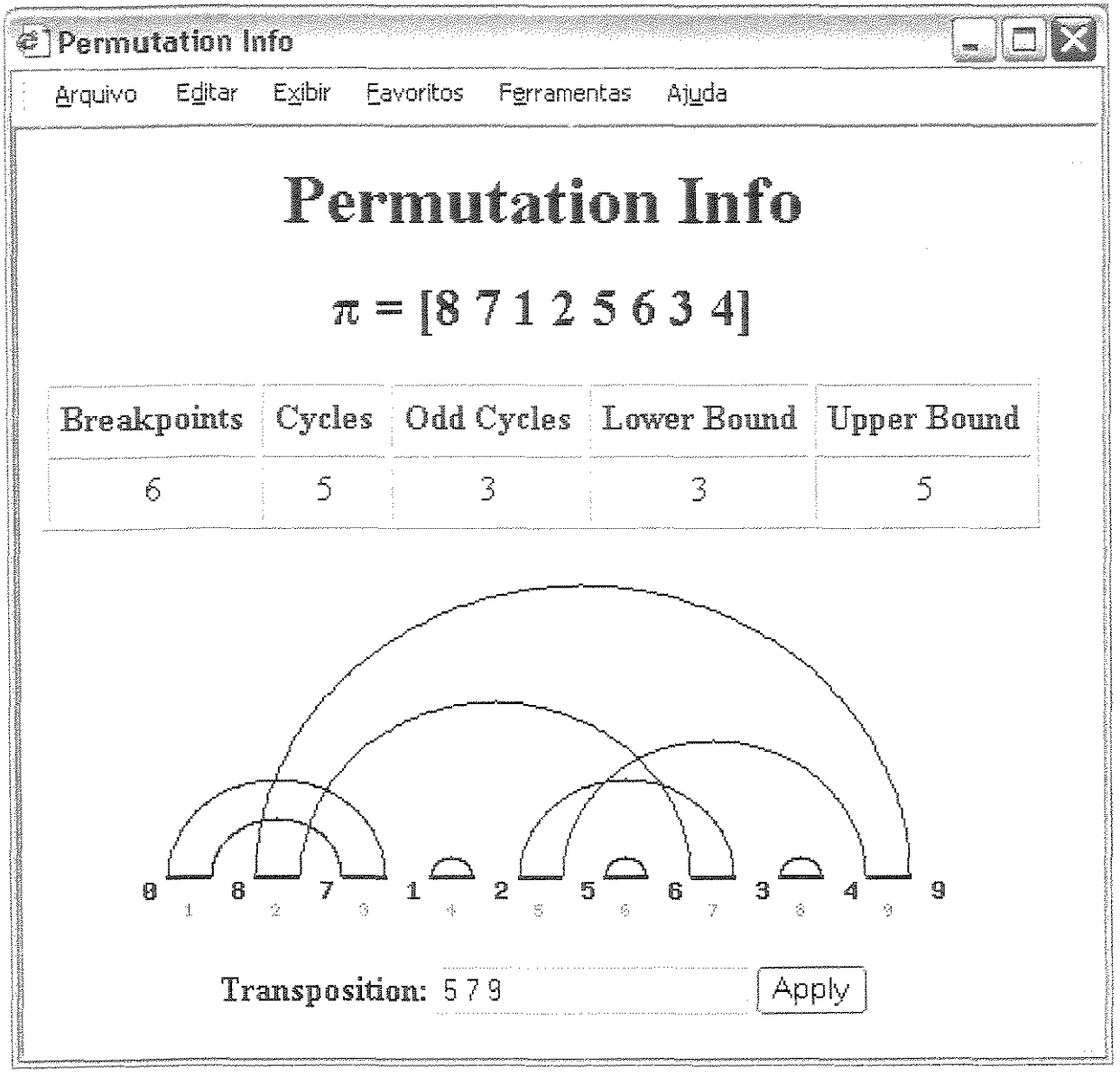

Figura A.3: *Permutation Info*: neste exemplo, vemos o "Diagrama de Ciclos" da permutação π = [8 7 1 2 5 6 3 4], formado por cinco ciclos, sendo três ciclos unitários (ímpares) e dois ciclos pares, antes que os blocos [5 6] e [3 4] fossem trocados de lugar pela transposição $\rho(5, 7, 9)$.

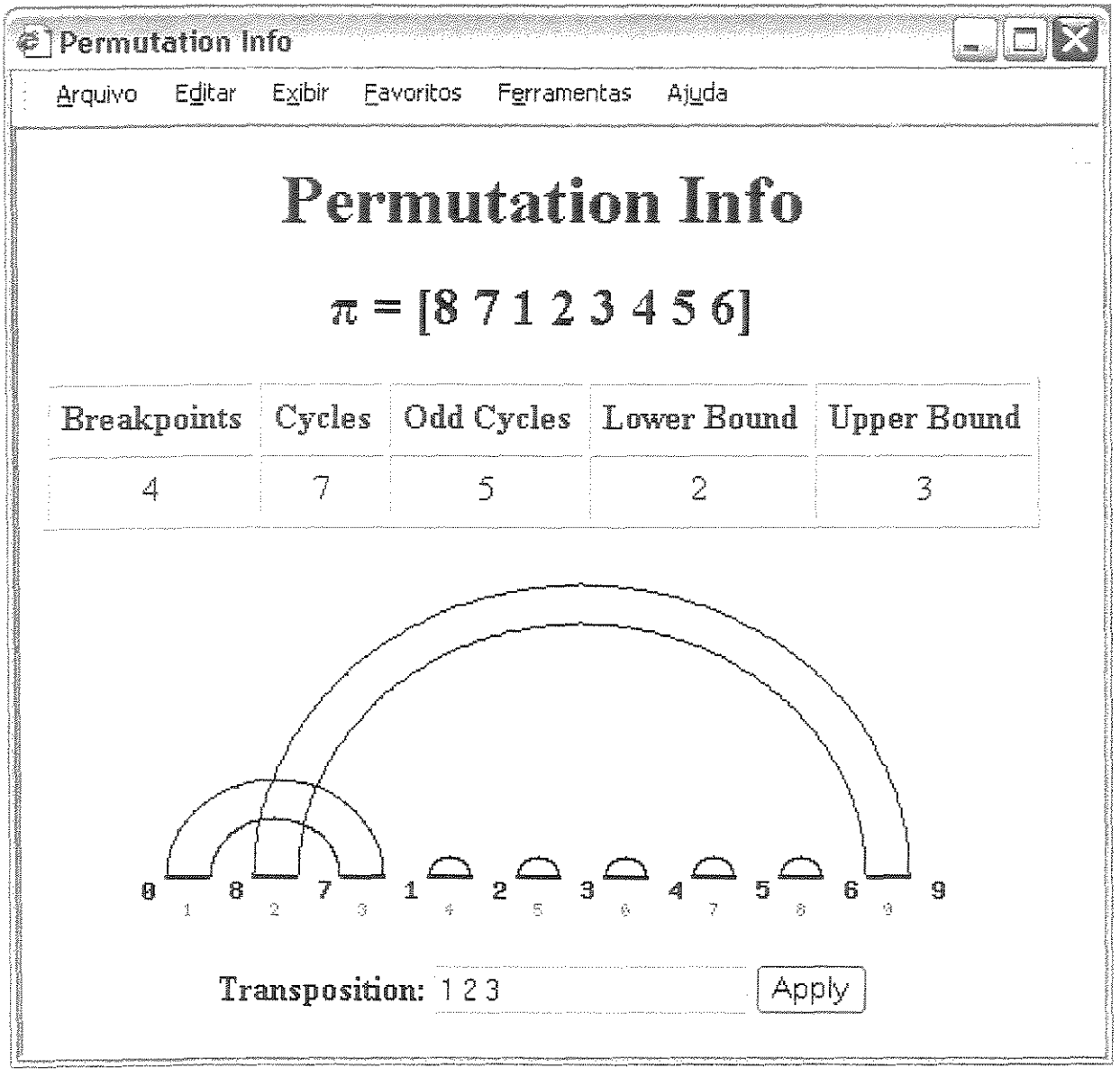

Figura A.4: *Permutation Info*: neste exemplo, vemos o "Diagrama de Ciclos" da permutação π = [8 7 1 2 3 4 5 6], formado por sete ciclos, sendo cinco ciclos unitários (ímpares) e dois ciclos pares, antes que os elementos 7 e 8 fossem trocados de lugar pela transposição $\rho(1, 2, 3)$.

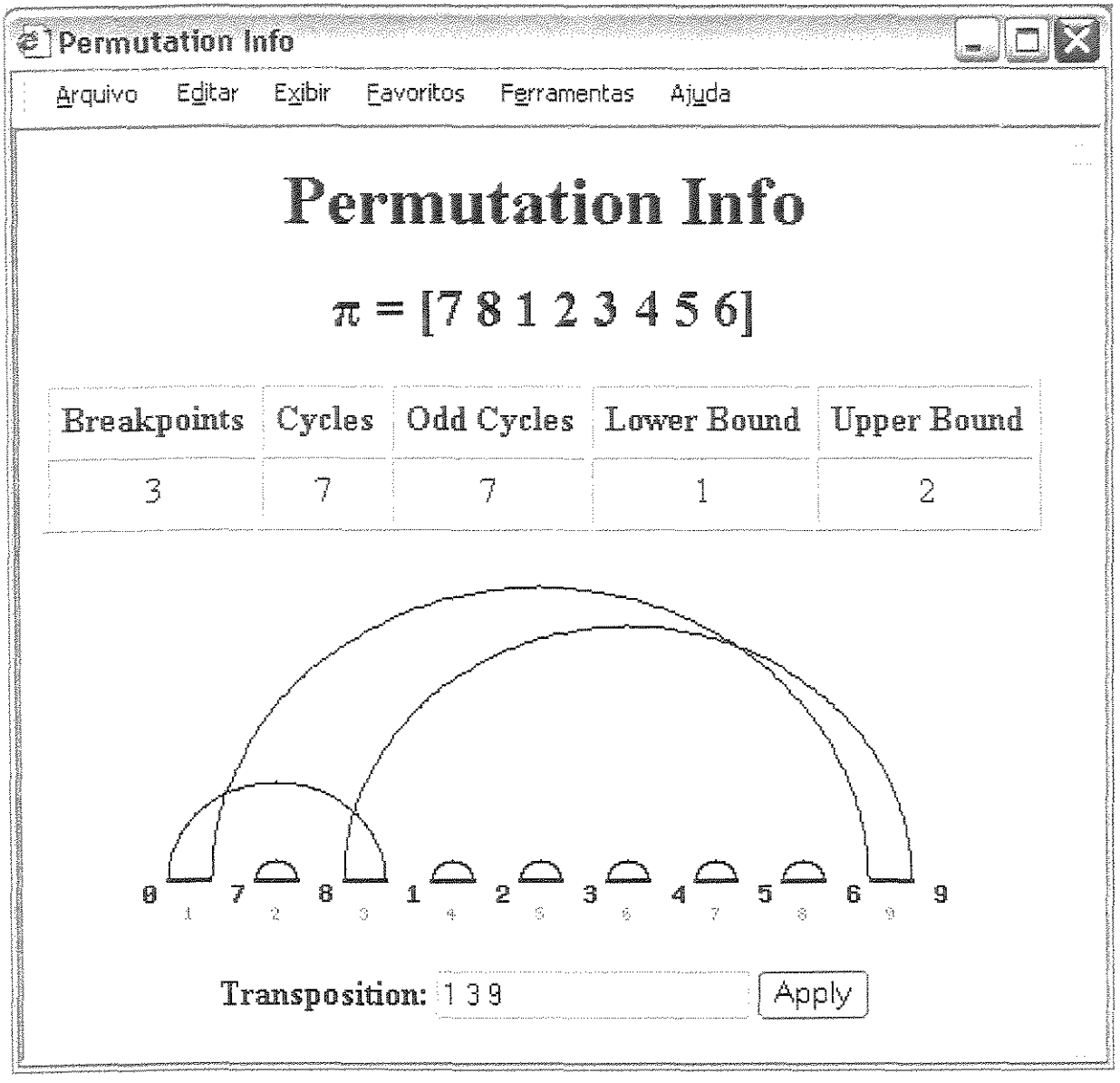

Figura A.5: *Permutation Info*: neste exemplo, vemos o "Diagrama de Ciclos" da permutação π = [7 8 1 2 3 4 5 6], formado por sete ciclos, sendo seis ciclos unitários e um ciclo ímpar longo, antes que os blocos [7 8] e [1 2 3 4 5 6] fossem trocados de lugar pela transposição $\rho(1, 3, 9)$.

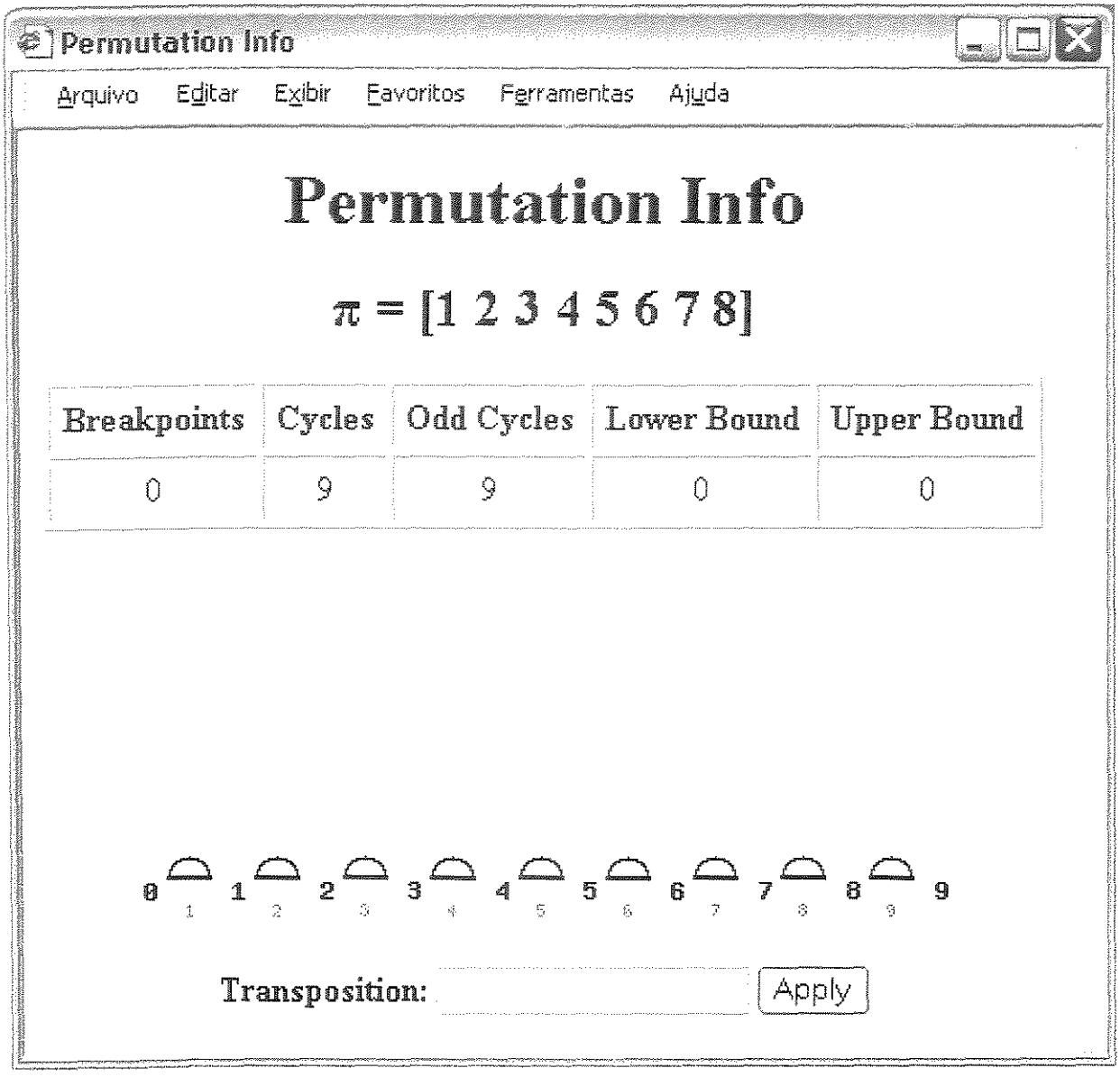

Figura A.6: *Permutation Info*: neste exemplo, vemos o "Diagrama de Ciclos" da permutação identidade $\iota_8 = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8]$, formado por nove ciclos unitários.

A.2.1 Distância Exata

Em vários momentos no decorrer desta tese, foi necessário obter o valor exato da distância de transposição de uma dada permutação. Infelizmente, até o presente momento não é conhecido nenhum algoritmo eficiente para o problema, nem uma prova que o mesmo seja NP-Difícil.

Assim, para testar as várias hipóteses sobre a distância de transposição exploradas nesta tese, resolvemos implementar um algoritmo exato utilizando a técnica de *branch and bound* como o descrito na Figura A.7.

```
ALGORITMO BRANCH AND BOUND PARA DISTÂNCIA DE TRANSPOSIÇÃO(π)
1  n ← |π|
2  best ← ⌈(3/4)(n + 1 − c_odd(π))⌉
3  current ← 0
4  Output: Transposition-Distance(π, best, current)

TRANSPOSITION-DISTANCE(π, best, current)
1  n ← |π|
2  if π = ι_n
3     then best ← current
4     else for all transpositions ρ
5          do π' ← ρπ
6             if current + (n+1−c_odd(π'))/2 < best
7                then best ← Transposition-Distance(π', best, current + 1)
8  Output: best
```

Figura A.7: Algoritmo *branch and bound* para o problema da distância de transposição. Calcula, de forma recursiva, o valor exato da distância de transposição ($d(\pi) = best$) sem retornar, explicitamente, uma série de transposições que permita transformar a permutação π na permutação identidade.

Apesar dos algoritmos que utilizam a técnica de *branch and bound* serem essencialmente exponenciais muitas vezes podem ser utilizados com sucesso para a solução de instâncias pequenas, mas muito úteis na prática.

Duas decisões são fundamentais para se conseguir uma bom desempenho de um algoritmo como o proposto aqui. Primeiro, devemos determinar um forma eficiente de expandir o espaço de busca à procura de uma solução ótima (*branch*). Segundo, temos que interromper uma busca tão logo ela se mostre infrutífera (*bound*).

Inicialmente para tentar não expandir desnecessariamente o espaço de busca utilizamos apenas transposições que não criassem novos *breakpoints*, já que, de acordo com o que foi provado por Christie [30], sempre existe uma série de transposições tal que, o número de *breakpoint*

nunca aumenta. Infelizmente na prática, esta versão se mostrou mais lenta devido ao tempo extra necessário para se calcular o novo número de *breakpoints* a cada transposição aplicada. Na nossa implementação, decidimos testar todas as permutações aplicáveis a uma certa permutação (linha 4 da parte recursiva).

Existem dois pontos no nosso algoritmo onde tentamos limitar o espaço de busca: na linha 2 da parte principal, quando calculamos uma estimativa inicial para a distância de transposição utilizando a *upper bound* de ciclos ímpares, e na linha 6 da parte recursiva, quando utilizamos a *lower bound* de ciclos ímpares para determinar se, naquele ponto da recursão, ainda é possível obter uma distância de transposição melhor do que a determinada pelo algoritmo até aquele momento.

Tanto a *lower bound*, quanto a *upper bound* de ciclos ímpares, foram definidas por Bafna e Pevzner [9]. Inicialmente, havíamos utilizado limites mais simples para o problema baseados nos números de *breakpoint*, mas logo percebemos que eles não eram suficientemente fortes para diminuir, de forma perceptível, o tempo de execução do algoritmo. Variações deste algoritmo foram utilizadas para o problema da distância de transposição de prefixos (ver, por exemplo, resultados da Tabela 9.1).

Em vários momentos, também estávamos interessados em calcular o diâmetro de transposição para permutações de um certo tamanho n. Usar o algoritmo exponencial da Figura A.7 para cada uma das $n!$ permutações de tamanho n seria totalmente inviável, mesmo para valores de n bem pequenos (por exemplo, $n = 10$).

Assim, resolvemos implementar uma variação do algoritmo de Bellman [10] para o problema de se calcular a distância de um vértice para todos os outros num grafo qualquer. Este algoritmo utiliza uma fila para garantir que cada aresta seja verificada um número constante de vezes. Este algoritmo é extremamente eficiente, calculando todas as distância em tempo linear no tamanho do grafo.

No nosso caso, cada vértice é uma permutação e existe uma aresta entre dois vértices π e σ, se existem transposições ρ' e ρ'', tal que $\pi = \rho'\sigma$ e $\sigma = \rho''\pi$. Estamos, então, interessados em calcular a distância neste grafo do vértice que representa a permutação identidade ι_n, para todos os outros vértices. Como temos que o número de arestas incidentes em cada vértice é da ordem de $O(n^3)$, então, a complexidade total do algoritmo é $O(n^3 n!)$.

Adaptamos também este algoritmo para o caso da distância de transposição de prefixos (ver Seção 9.4.1). O grande limitante deste algoritmo, como demonstram os testes exibidos naquela seção, é a sua enorme necessidade de memória, por exemplo, seriam necessários aproximadamente 30GB de memória física para que o algoritmo calculasse o diâmetro de transposição (ou de transposição de prefixos) para permutações de tamanho $n = 12$.

```
CÁLCULO DA DISTÂNCIA DE TRANSPOSIÇÃO PARA TODAS AS PERMUTAÇÕES(n)
 1  for all permutations π of size n
 2  do d(π) = ∞
 3  d(ι_n) ← 0
 4  Queue ← ι_n
 5  While Queue not empty
 6  do π ← Pop(Queue)
 7     for all transpositions ρ
 8     do π' ← ρπ
 9        if d(π') = ∞
10          then d(π') ← d(π) + 1
11               Push(Queue, π')
12  Output: d(π), for all permutations π of size n
```

Figura A.8: Algoritmo desenvolvido para calcular a distância de transposição para todas as permutação de um certo tamanho n.

A.2.2 Heurística

Logo no início dos trabalhos desta tese, sentimos a necessidade de poder contar com um algoritmo polinomial, de fácil implementação, que fornecesse uma boa aproximação para o valor da distância de transposição.

A princípio, pensamos em implementar o algoritmo proposto por Bafna e Pevzner [9], mas este algoritmo é complexo demais para nossos propósitos, sendo, inclusive, objeto de trabalhos de mestrado [41]. Como estávamos interessados apenas na distância de transposição e não em obter uma série de transposições que transformasse uma permutação qualquer na permutação identidade, resolvemos implementar uma heurística gulosa para o problema.

O algoritmo da Figura A.9 funciona da seguinte maneira: enquanto a permutação de entrada não tiver sido ordenada e ainda não tiverem sido aplicadas tantas transposições quanto o número indicado pela *upper bound* de ciclos ímpares definida por Bafna e Pevzner, teste todas as transposições e aplique aquela com o maior *Score*. Definimos o *Score* de uma transposição ρ com relação a permutação π, como o número de ciclos que ρ cria ($\Delta c(\pi, \rho)$), mais o número de novos ciclos ímpares ($\Delta c_{odd}(\pi, \rho)$) e mais o número de novos ciclos bons ($\Delta c_{good}(\pi, \rho)$). Um ciclo é chamado de bom, se ele pode ser quebrado em três ciclos ímpares menores com apenas uma transposição.

Executamos a nossa heurística para o problema da distância de transposição para todas as permutações com tamanho menor ou igual a 11, totalizando aproximadamente 40 milhões de casos de testes. Os resultados foram satisfatórios: em cerca de 98% dos casos a nossa heurística alcançou o valor exato da distância de transposição. Na Figura A.1, comparamos a

nossa heurística com o algoritmo WDM'2000 proposto por Walter, Dias e Meidanis [121].

HEURÍSTICA PARA O PROBLEMA DA DISTÂNCIA DE TRANSPOSIÇÃO(π)

1 $d \leftarrow 0$
2 $n \leftarrow |\pi|$
3 $Max \leftarrow \left\lceil \frac{3}{4}(n + 1 - c_{odd}(\pi)) \right\rceil$
4 **While** $\pi \neq \iota_n$ and $d < Max$
5 **do** $Score' \leftarrow -\infty$
6 **for** all transpositions ρ
7 **do** $Score \leftarrow \Delta c(\pi, \rho) + \Delta c_{odd}(\pi, \rho) + \Delta c_{good}(\pi, \rho)$
8 **if** $Score > Score'$
9 **then** $Score' \leftarrow Score$
10 $\rho' \leftarrow \rho$
11 $\pi \leftarrow \rho'\pi$
12 $d \leftarrow d + 1$
13 Output: d

Figura A.9: Heurística gulosa desenvolvida para o problema da distância de transposição.

O algoritmo WDM'2000 é um algoritmo de aproximação com fator 2.25 para o problema da distância de transposição com complexidade $O(n^2)$ e que, diferentemente da heurística apresentada neste apêndice, pode ser usado para se obter, além do valor da distância de transposição, uma série de transposições que efetivamente transformem uma permutação qualquer na permutação identidade. Outra desvantagem da nossa heurística é sua alta complexidade assintótica, $O(n^5)$, já que temos que testar, em cada passo, todas as $O(n^3)$ possíveis transposições e levamos tempo linear para calcular o *Score* de cada transposição.

Por último, é importante destacar que o passo 4 da Figura A.9 é usado para interromper o algoritmo caso uma solução não tenha sido obtida usando menos transposições que a *upper bound* de ciclos ímpares. Logo, esta heurística nunca apresenta um fator de aproximação maior que $\frac{3}{2}$, mesmo quando ela não é capaz de determinar uma série de transposições que garanta esta aproximação.

Tentamos obter um método guloso mais eficiente para a escolha da transposição a ser aplicada a cada passo. Infelizmente, mesmo após várias experiências, não conseguimos uma variação deste algoritmo que fornecesse resultados melhores do que aqueles apresentados na Tabela A.1.

Todos os arquivos relacionados a esta tese, incluindo os programas apresentados neste apêndice, podem ser obtidos diretamente com o aluno ou com seu orientador.

	Heurística		WMD'2000	
N	Diferença	Aproximação	Diferença	Aproximação
1	0	1	0	1
2	0	1	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	0	1	0,83%	1,33
7	0,02%	1,25	1,42%	1,50
8	0,08%	1,25	2,89%	1,75
9	0,15%	1,25	3,95%	1,75
10	1,11%	1,50	5,88%	1,75
11	2,18%	1,50	7,19%	2,00

Tabela A.1: Comparação entre nossa heurística para o problema da distância de transposição (Figura A.9) e o algoritmo WDM'2000 proposto por Walter, Dias e Meidanis [121]. As colunas rotuladas de "Diferença" indicam as porcentagens das permutações em que os algoritmos testados fornecem uma distância maior do que a distância exata, de acordo com os resultados obtidos usando o algoritmo da Figura A.8. Já as colunas denominadas "Aproximação" indicam o fator de aproximação efetivamente obtido por cada algoritmo.

Bibliografia

[1] M. D. Adams, S. E. Celniker, R. A. Holt, C. A. Evans, J. D. Gocayne, P. G. Amanatides, S. E. Scherer, P. W. Li, R. A. Hoskins, R. F. Galle, R. A. George, S. E. Lewis, S. Richards, M. Ashburner, S. N. Henderson, G. G. Sutton, J. R. Wortman, M. D. Yandell, Q. Zhang, L. X. Chen, R. C. Brandon, Y.-H. C. Rogers, R. G. Blazej, M. Champe, B. D. Pfeiffer, K. H. Wan, C. Doyle, E. G. Baxter, G. Helt, C. R. Nelson, G. L. G. Miklos, J. F. Abril, A. Agbayani, H.-J. An, C. Andrews-Pfannkoch, D. Baldwin, R. M. Ballew, A. Basu, J. Baxendale, L. Bayraktaroglu, E. M. Beasley, K. Y. Beeson, P. V. Benos, B. P. Berman, D. Bhandari, S. Bolshakov, D. Borkova, M. R. Botchan, J. Bouck, P. Brokstein, P. Brottier, K. C. Burtis, D. A. Busam, H. Butler, E. Cadieu, A. Center, I. Chandra, J. M. Cherry, S. Cawley, C. Dahlke, L. B. Davenport, P. Davies, B. de Pablos, A. Delcher, Z. Deng, A. D. Mays, I. Dew, S. M. Dietz, K. Dodson, L. E. Doup, M. Downes, S. Dugan-Rocha, B. C. Dunkov, P. Dunn, K. J. Durbin, C. C. Evangelista, C. Ferraz, S. Ferriera, W. Fleischmann, C. Fosler, A. E. Gabrielian, N. S. Garg, W. M. Gelbart, K. Glasser, A. Glodek, F. Gong, J. H. Gorrell, Z. Gu, P. Guan, M. Harris, N. L. Harris, D. Harvey, T. J. Heiman, J. R. Hernandez, J. Houck, D. Hostin, K. A. Houston, T. J. Howland, M.-H. Wei, C. Ibegwam, M. Jalali, F. Kalush, G. H. Karpen, Z. Ke, J. A. Kennison, K. A. Ketchum, B. E. Kimmel, C. D. Kodira, C. Kraft, S. Kravitz, D. Kulp, Z. Lai, P. Lasko, Y. Lei, A. A. Levitsky, J. Li, Z. Li, Y. Liang, X. Lin, X. Liu, B. Mattei, T. C. McIntosh, M. P. McLeod, D McPherson, G. Merkulov, N. V. Milshina, C. Mobarry, J. Morris, A. Moshrefi, S. M. Mount, M. Moy, B. Murphy, L. Murphy, D. M. Muzny, D. L. Nelson, D. R. Nelson, K. A. Nelson, K. Nixon, D. R. Nusskern, J. M. Pacleb, M. Palazzolo, G. S. Pittman, S. Pan, J. Pollard, V. Puri, M. G. Reese, K. Reinert, K. Remington, R. D. C. Saunders, F. Scheeler, H. Shen, B. C. Shue, I. Sidén-Kiamos, M. Simpson, M. P. Skupski, T. Smith, E. Spier, A. C. Spradling, M. Stapleton, R. Strong, E. Sun, R. Svirskas, C. Tector, R. Turner, E. Venter, A. H. Wang, X. Wang, Z.-Y. Wang, D. A. Wassarman, G. M. Weinstock, J. Weissenbach, S. M. Williams, T. Woodage, K. C. Worley, D. Wu, S. Yang, Q. A. Yao, J. Ye, R.-F. Yeh, J. S. Zaveri, M. Zhan, G. Zhang, Q. Zhao, L. Zheng, X. H. Zheng, F. N. Zhong, W. Zhong, X. Zhou, S. Zhu, X. Zhu, H. O. Smith, R. A. Gibbs, E. W. Myers, G. M. Rubin, and J. C. Venter. The genome sequence of *Drosophila melanogaster*. *Science*, 287(5461):2185–2195, March 2000.

[2] A. Aggarwal and T. Leighton. A tight lower bound for the train reversal problem. *Information Processing Letters*, 35(6):301–304, September 1990.

[3] M. Aigner and D. B. West. Sorting by insertion of leading element. *Journal of Combinatorial Theory*, 45:306–309, 1987.

[4] N. Amato, M. Blum, S. Irani, and R. Rubinfeld. Reversing trains: A turn of the century sorting problem. *Journal of Algorithms*, 10(3):413–428, September 1989.

[5] D. A. Bader, B. M. E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[6] D. A. Bader, B. M. E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. In *Proceedings of the Seventh Workshop on Algorithms and Data Structures (WADS'01)*. Springer Verlag, 2001.

[7] V. Bafna and P. A. Pevzner. Sorting by transpositions. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 614–623, San Francisco, USA, January 1995.

[8] V. Bafna and P. A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25(2):272–289, 1996.

[9] V. Bafna and P. A. Pevzner. Sorting by transpositions. *SIAM Journal on Discrete Mathematics*, 11(2):224–240, May 1998.

[10] R. E. Bellman. On a rouiting problem. *Quaterly of Applied Mathematics*, 16:87–90, 1958.

[11] A. Bergeron. A very elementary presentation of the hannenhalli-pevzner theory. In *Proceedings of the 12th Annual Symposium of the Combinatorial Pattern Matching (CPM'2001)*, volume 2089 of *Lecture Notes in Computer Science*, pages 106–117, Berlin, Germany, September 2001. Springer-Verlag.

[12] A. Bergeron, S. Heber, and J. Stoye. Common intervals and sorting by reversals: A marriage of necessity. *Bioinformatics*, 18:S1–S10, 2002.

[13] P. Berman and S. Hannenhalli. Fast sorting by reversal. In D. S. Hirschberg and E. W. Myers, editors, *Proceedings of Combinatorial Pattern Matching (CPM'96), 7th Annual Symposium*, volume 1075 of *Lecture Notes in Computer Science*, pages 168–185, Laguna Beach, USA, January 1996. Springer.

[14] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In *Proceedings of the 10th European Symposium on Algorithms (ESA'2002)*, Lecture Notes in Computer Science, Rome, Italy, September 2002. Springer.

[15] P. Berman and M. Karpinski. On some tighter inapproximability results (extended abstract). In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming (ICALP'99)*, volume 1644 of *Lecture Notes in Computer Science*, pages 200–209. Springer, 1999.

[16] M. Blanchette, T. Kunisawa, and D. Sankoff. Parametric genome rearrangement. *Journal of Computational Biology*, 172:11–17, 1996.

[17] Brazilian Genome (BrGene) – The Virtual Institute of Genomic Research, September 2002. www.brgene.lncc.br.

[18] A. Caprara. Sorting by reversals is difficult. In *Proceedings of the First International Conference on Computational Molecular Biology - (RECOMB'97)*, pages 75–83, New York, USA, January 1997. ACM Press.

[19] A. Caprara. Formulations and hardness of multiple sorting by reversals. In S. Istrail, P. A. Pevzner, and M. Waterman, editors, *Proceedings of the 3rd Annual International Conference on Computational Molecular Biology (RECOMB'99)*, pages 84–93, Lyon, France, 1999. ACM Press.

[20] A. Caprara. On the tightness of the alternating-cycle lower bound for sorting by reversals. *Journal of Combinatorial Optimization*, 3:149–182, 1999.

[21] A. Caprara. Sorting permutations by reversals and eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics*, 12(1):91–110, February 1999.

[22] A. Caprara and G. Lancia. Experimental and statistical analysis of sorting by reversals. In D. Sankoff and J. H. Nadeau, editors, *Comparative Genomics: Empirical and Analyitical Approaches to Gene Order Dynamics, Map Alignment and Evolution of Gene Families*. Kluwer Academic Publishers, Le Chantecler, Canada, September 2000.

[23] A. Caprara, G. Lancia, and S.-K. Ng. A column-generation based branch-and-bound algorithm for sorting by reversals. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, The American Mathematical Society*, 47:213–226, 1999.

[24] A. Caprara, G. Lancia, and S.-K. Ng. Sorting permutations by reversals through branch-and-price. Technical Report OR-99-1, DEIS - Operations Research Group, University of Bologna, 1999.

[25] A. Caprara, G. Lancia, and S.-K. Ng. Fast practical solution of sorting by reversals. In *Proceedings of the 11th ACM-SIAM Annual Symposium on Discrete Algorithms - (SODA'00)*, pages 12–21, San Francisco, USA, 2000. ACM Press.

[26] F. R. Cerqueira. Montagem de fragmentos de DNA. Master's thesis, University of Campinas, Brazil, 2000. In Portuguese.

[27] T. Chen and S. S. Skiena. Sorting with fixed-length reversals. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 71:269–295, 1996.

[28] D. A. Christie. Sorting permutations by block-interchanges. *Information Processing Letters*, 60(4):165–169, November 1996.

[29] D. A. Christie. A 3/2-approximation algorithm for sorting by reversals. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 244–252, San Francisco, USA, January 1998.

[30] D. A. Christie. *Genome Rearrangement Problems*. PhD thesis, Glasgow University, 1998.

[31] CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico, September 2002. www.cnpq.br.

[32] The Genome International Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[33] Copersucar – Cooperativa de Produtores de Cana, Açúcar e Álcool do Estado de São Paulo, September 2002. www.copersucar.com.br.

[34] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[35] CPAN – Comprehensive Perl Archive Network, September 2002. www.cpan.org.

[36] L. R. A. F. Curado. Problema da ordenação por transposições em rearranjos de genomas. Master's thesis, University of Brasília, Brazil, 2001. In Portuguese.

[37] A. C. R. da Silva, J. A. Ferro, F. C. Reinach, C. S. Farah, L. R. Furlan, R. B. Quaggio, C. B. Monteiro-Vitorello, M. A. Van Sluys, N. F. Almeida, L. M. C. Alves, A. M. do Amaral, M. C. Bertolini, L. E. A. Camargo, G. Camarotte, F. Cannavan, J. Cardozo, F. Chambergo, L. P. Ciapina, R. M. B. Cicarelli, L. L. Coutinho, J. R. Cursino-Santos,

H. El-Dorry, J. B. Faria, A. J. S. Ferreira, R. C. C. Ferreira, M. I. T. Ferro, E. F. Formighieri, M. C. Franco, C. C. Greggio, A. Gruber, A. M. Katsuyama, L. T. Kishi, R. P. Leite, E. G. M. Lemos, M. V. F. Lemos, E. C. Locali, M. A. Machado, A. M. B. N. Madeira, N. M. Martinez-Rossi, E. C. Martins, J. Meidanis, C. F. M. Menck, C. Y. Miyaki, D. H. Moon, L. M. Moreira, M. T. M. Novo, V. K. Okura, M. C. Oliveira, V. R. Oliveira, H. A. Pereira, A. Rossi, J. A. D. Sena, C. Silva, R. F. de Souza, L. A. F. Spinola, M. A. Takita, R. E. Tamura, E. C. Teixeira, R. I. D. Tezza, M. Trindade dos Santos, D. Truffi, S. M. Tsai, F. F. White, J. C. Setubal, and J. P. Kitajima. Comparison of the genomes of two *Xanthomonas* pathogens with differing host specificities. *Nature*, 417(6887):459–463, May 2002.

[38] B. DasGupta, T. Jiang, S. Kannan, M. Li, and E. Sweedyk. On the complexity and approximation of syntenic distance. *Discrete Applied Mathematics, Second Special Issue on Computational Biology*, 88:59–82, 1998.

[39] dbEST – The International Expressed Sequence Tags Database, September 2002. www.ncbi.nlm.nih.gov/dbEST.

[40] F. E. S. de Araújo. Rearranjo de genomas por reversões. Master's thesis, University of São Paulo, Brazil, 1998. In Portuguese.

[41] E. T. G. de Oliveira. Implementaçao de algoritmos para o problema de ordenação de transposições. Master's thesis, University of Brasília, Brazil, 2001. In Portuguese.

[42] Z. Dias and J. Meidanis. Genome rearrangements distance by fusion, fission, and transposition is easy. In *Proceedings of the String Processing and Information Retrieval (SPIRE'2001)*, pages 250–253, Laguna de San Rafael, Chile, November 2001. IEEE Computer Society.

[43] Z. Dias and J. Meidanis. The genome rearrangement distance problem with arbitrary weights. Technical Report IC-02-01, Institute of Computing - University of Campinas, March 2002.

[44] Z. Dias and J. Meidanis. Sorting by prefix transpositions. In A. H. F. Laender and A. L. Oliveira, editors, *Proceedings of the String Processing and Information Retrieval (SPIRE'2002)*, number 2476 in Lecture Notes in Computer Science, pages 65–76, Lisboa, Portugal, September 2002. Springer-Verlag, Berlin.

[45] H. Dweighter. *American Mathematical Monthly*, volume 82, page 1010. The Mathematical Association of America, 1975.

[46] The C. elegans Sequencing Consortium. Genome sequence of the nematode c. elegans: A platform for investigating biology. *Science*, 282(5396):2012–2018, 1999.

[47] FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo, September 2002. www.fapesp.br.

[48] V. Ferretti, J. H. Nadeau, and D. Sankoff. Original synteny. In D. S. Hirschberg and E. W. Myers, editors, *Proceedings of Combinatorial Pattern Matching (CPM'96), 7th Annual Symposium*, volume 1075 of *Lecture Notes in Computer Science*, pages 159–167, Laguna Beach, USA, January 1996. Springer.

[49] D. Frishmana, A. Mironov, and M. Gelfand. Starts of bacterial genes: Estimating the reliability of computer predictions. *Gene*, 234:257–265, 1999.

[50] *Xylella* – Genoma Funcional, September 2002. www.lbm.fcav.unesp.br/fun.

[51] Fundecitrus – Fundo de Defesa da Citricultura, October 2002. www.fundecitrus.com.br.

[52] W. H. Gates and C. H. Papadimitriou. Bounds for sorting by prefix reversals. *Discrete Mathematics*, 27:47–57, 1979.

[53] GenBank, September 2002. www.ncbi.nlm.nih.gov/Genbank.

[54] P. Green. Phrap documentation, September 2002. www.phrap.org.

[55] Q.-P. Gu, S. Peng, and H. Sudborough. A 2-approximation algorithm for genome rearrangements by reversals and transpositions. *Theoretical Computer Science*, 210(2):327–339, January 1999.

[56] Q.-P. Gu, S. Peng, and H. Sudbourough. Approximating algorithms for genome rearrangements. In *Proceedings of the 7th Workshop on Genome Informatics (GIW'96)*, Tokyo, Japan, December 1996.

[57] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[58] S. A. Guyer, L. S. Heath, and J. P. Vergara. Subsequence and run heuristics for sorting by transpositions. In *Proceedings of the 4th DIMACS International Algorithm Implementation Challenge*, August 1995.

[59] S. Hannenhalli. *Transforming Men into Mice*. PhD thesis, Pennsylvania State University, 1995.

[60] S. Hannenhalli. Polynomial-time algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics*, 71:137–151, 1996.

[61] S. Hannenhalli, C. Chappey, E. V. Koonin, and P. A. Pevzner. Genome sequence comparison and scenarios for gene rearrangements: a test case. *Genomics*, 30:299–311, 1995.

[62] S. Hannenhalli and P. A. Pevzner. Transforming men into mice (polynomial algorithm for genomic distance problem). In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS'95)*, pages 581–592, Los Alamitos, USA, October 1995. IEEE Computer Society Press.

[63] S. Hannenhalli and P. A. Pevzner. To cut... or not to cut (applications of comparative physical maps in molecular evolution). In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 304–313, Atlanta, USA, January 1996.

[64] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM*, 46(1):1–27, January 1999.

[65] L. S. Heath and J. P.Vergara. Sorting by bounded block-moves. *Discrete Applied Mathematics, Second Special Issue on Computational Biology*, 88:181–206, 1998.

[66] M. H. Heydari and I. H. Sudborough. Sorting by prefix reversals is NP-complete. To be submitted.

[67] M. H. Heydari and I. H. Sudborough. On the diameter of the pancake network. *Journal of Algorithms*, 25:67–94, 1997.

[68] The Arabidopsis Initiative. Analysis of the genome sequence of the flowering plant *Arabidopsis thaliana*. *Nature*, 408:796–815, December 2000.

[69] R. W. Irving and D. A. Christie. Sorting by reversals: on a conjecture of kececioglu and sankoff. Technical Report TR-95-12, Department of Computing Science, University of Glasgow, May 1995.

[70] N. Jacobson. *Basic Algebra*. W. H. Freeman and Company, New York, 1985.

[71] M. Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36:265–289, 1985.

[72] H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 344–351, New Orleans, USA, January 1997.

[73] H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal on Computing*, 29(3):880–892, January 2000.

[74] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals. Technical Report 1824, Centre de Recherches mathématiques, Université de Montréal, July 1992.

[75] J. D. Kececioglu and R. Ravi. Of mice and men: Algorithms for evolutionary distances between genomes with translocation. In *Proceedings of the 6th Annual Symposium on Discrete Algorithms*, pages 604–613, New York, USA, January 1995. ACM Press.

[76] J. D. Kececioglu and D. Sankoff. Exact and approximation algorithms for the inversion distance between two chromosomes. In A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, editors, *Proceedings of the 4th Annual Symposium of the Combinatorial Pattern Matching (CPM'93)*, volume 684 of *Lecture Notes in Computer Science*, pages 87–105, Padova, Italy, June 1993. Springer-Verlag.

[77] J. D. Kececioglu and D. Sankoff. Efficient bounds for oriented chromosome inversion distance. In M. Crochemore and D. Gusfield, editors, *Proceedings of the 5th Annual Symposium of the Combinatorial Pattern Matching (CPM'94)*, volume 807 of *Lecture Notes in Computer Science*, pages 307–325, Asilomar, USA, June 1994. Springer.

[78] J. D. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13:180–210, January 1995.

[79] J. Kleinberg and D. Liben-Nowell. The syntenic diameter of the space of n-chromosome genomes. In D. Sankoff and J. H. Nadeau, editors, *Comparative Genomics: Empirical and Analyitical Approaches to Gene Order Dynamics, Map Alignment and Evolution of Gene Families*. Kluwer Academic Publishers, Le Chantecler, Canada, September 2000.

[80] Laboratório de BioInformática (LBI), September 2002. www.lbi.ic.unicamp.br.

[81] L. T. Li. Montagem de fragmentos de DNA pelo método "Ordered Shotgun Sequencing" (OSS). Master's thesis, University of Campinas, Brazil, 2001. In Portuguese.

[82] D. Liben-Nowell. On the structure of syntenic distance. *Journal of Computational Biology*, 8(1), February 2001.

[83] D. Liben-Nowell. Gossip is synteny: Incomplete gossip and the syntenic distance between genomes. *Journal of Algorithms*, 43(2), May 2002.

[84] D. Liben-Nowell and J. Kleinberg. Structural properties and tractability results for linear synteny. In *Proceedings of the 11th Annual Symposium of the Combinatorial Pattern Matching (CPM'2000)*. Springer Verlag, 2000.

[85] G.-H. Lin and G. Xue. Signed genome rearrangement by reversals and transpositions: Models and approximations. In *Fifth Annual International Computing and Combinatorics Conference (COCOON'99)*, pages 71–80, 1999.

[86] LICR – Ludwig Institute for Cancer Research, September 2002. www.ludwig.org.br.

[87] S. MacLane and G. Birkhoff. *Algebra*. The Macmillan Company, London, sixth printing edition, 1971.

[88] Medalha de Mérito Científico e Tecnológico. Mário Covas. Decretos Lei número 44716 e 44717. *Diário Oficial do Estado de São Paulo*, 18 de fevereiro de 2000.

[89] J. Meidanis and Z. Dias. An alternative algebraic formalism for genome rearrangements. In D. Sankoff and J. H. Nadeau, editors, *Proceedings of the Gene Order Dynamics, Comparative Maps and Multigene Families (DCAF'2000)*, Le Chantecler, Canada, September 2000.

[90] J. Meidanis and Z. Dias. An alternative algebraic formalism for genome rearrangements. In D. Sankoff and J. H. Nadeau, editors, *Comparative Genomics: Empirical and Analyitical Approaches to Gene Order Dynamics, Map Alignment and Evolution of Gene Families*, pages 213–223. Kluwer Academic Publishers, November 2000.

[91] J. Meidanis and Z. Dias. Genome rearrangements distance by fusion, fission, and transposition is easy. Technical Report IC-01-07, Institute of Computing - University of Campinas, July 2001.

[92] J. Meidanis, M. E. M. T. Walter, and Z. Dias. Distância de reversão de cromossomos circulares. In *Proceedings of the XXIV Seminário Integrado de Software e Hardware (SEMISH'97)*, pages 119–131, August 1997. In Portuguese.

[93] J. Meidanis, M. E. M. T. Walter, and Z. Dias. Transposition distance between a permutation and its reverse. In R. Baeza-Yates, editor, *Proceedings of the 4th South American Workshop on String Processing (WSP'97)*, pages 70–79, Valparaiso, Chile, 1997. Carleton University Press.

[94] J. Meidanis, M. E. M. T. Walter, and Z. Dias. A lower bound on the reversal and transposition diameter. Technical Report IC-00-16, Institute of Computing - University of Campinas, October 2000.

[95] J. Meidanis, M. E. M. T. Walter, and Z. Dias. Reversal distance of signed circular chromosomes. Technical Report IC-00-23, Institute of Computing - University of Campinas, December 2000.

[96] J. Meidanis, M. E. M. T. Walter, and Z. Dias. A lower bound on the reversal and transposition diameter. *Journal of Computational Biology*, 9(5), 2002. Submitted: 14/January/2000. Acepted: 16/July/2002.

[97] R. T. Miller, A. G. Christoffels, C. Gopalakrishman, J. A. Burke, A. A. Ptitsyn, T. R. Broveak, and W. A. Hide. A comprehensive approach to clustering of expressed human gene sequence: The sequence tag alignment and consensus knowledge base. *Genome Research*, 9:1143–1155, 1999.

[98] E. W. Myers. A whole-genome assembly of *Drosophila*. *Science*, 287:2196–2204, 2000.

[99] V. K. Okura. Bioinformática de projetos genoma de bactérias. Master's thesis, University of Campinas, Brazil, 2002. In Portuguese.

[100] Rede ONSA – Organization for Nucleotide Sequencing and Analysis, September 2002. www.watson.fapesp.br/onsa/onsagera.htm.

[101] J. D. Palmer and L. A. Herbon. Plant mitochondrial dna evolves rapidly in structure, but slowly in sequence. *Journal of Molecular Evolution*, 27:87–97, 1988.

[102] J. D. Palmer, B. Osorio, and W. F. Thompson. Evolutionary significance of inversions in legume chloroplast dnas. *Current Genetics*, 14:65–74, 1988.

[103] P. A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. The MIT Press, 2000.

[104] *Schistosoma mansoni* EST Genome Project, September 2002. verjo18.iq.usp.br/schisto.

[105] J. C. Setubal and J. Meidanis. *Uma Introdução à Biologia Computacional*. CIP, 1994. In Portuguese.

[106] J. C. Setubal and J. Meidanis. *Introduction to Computional Molecular Biology*. PWS Publishing Company, 1997.

[107] J. C. Setubal and R. F. Werneck. A program for buiding contig scaffolds in double-barreled shotgun genome sequencing. Technical Report IC-00-20, Institute of Computing - University of Campinas, December 2000.

[108] A. J. G. Simpson, F.C. Reinach, P. Arruda, F. A. Abreu, M. Acencio, R. Alvarenga, L. M. C. Alves, J. E. Araya, G. S. Baia, C. S. Baptista, M. H. Barros, E. D. Bonaccorsi, S. Bordin, J. M. Bove, M. R. S. Briones, M. R. P. Bueno, A. A. Camargo, L. E. A. Camargo, D. M. Carraro, H. Carrer, N. B. Colauto, C. Colombo, F. F. Costa, M. C. R. Costa, C. M. Costa-Neto, L. L. Coutinho, M. Cristofani, E. Dias-Neto, C. Docena, H. El-Dorry, A. P. Facincani, A. J. S. Ferreira, V. C. A. Ferreira, J. A. Ferro, J. S. Fraga, S. C. Fran?a, M. C. Franco, M. Frohme, L. R. Furlan, M. Garnier, G. H. Goldman, M. H. S. Goldman, S. L. Gomes, A. Gruber, P. L. Ho, J. D. Hoheisel, M. L. Junqueira, E. L. Kemper, J. P. Kitajima, J. E. Krieger, E. E. Kuramae, F. Laigret, M. R. Lambais, L. C. C. Leite, E. G. M. Lemos, M. V. F. Lemos, S. A. Lopes, C. R. Lopes, J. A. Machado, M. A. Machado, A. M. B. N. Madeira, H. M. F. Madeira, C. L. Marino, M. V. Marques, E. A. L. Martins, E. M. F. Martins, A. Y. Matsukuma, C. F. M. Menck, E. C. Miracca, C. Y. Miyaki, C. B. Monteiro-Vitorello, D. H. Moon, M. A. Nagai, A. L. T. O. Nascimento, L. E. S. Netto, A. Nhani, F. G. Nobrega, L. R. Nunes, M. A. Oliveira, M. C. De Oliveira, R. C. De Oliveira, D. A. Palmieri, A. Paris, B. R. Peixoto, G. A. G. Pereira, H. A. Pereira, J. B. Pesquero, R. B. Quaggio, P. G. Roberto, V. Rodrigues, A. J. De M. Rosa, V. E. De Rosa, R. G. De Sá, R. V. Santelli, H. E. Sawasaki, A. C. R. Da Silva, A. M. Da Silva, F. R. Da Silva, W. A. Silva, J. F. Da Silveira, M. L. Z. Silvestri, W. J. Siqueira, A. A. De Souza, A. P. De Souza, M. F. Terenzi, D. Truffi, S. M. Tsai, M. H. Tsuhako, H. Vallada, M. A. Van Sluys, S. Verjovski-Almeida, A. L. Vettore, M. A. Zago, M. Zatz, J. Meidanis, and J. C. Setubal. The genome sequence of the plant pathogen *Xylella fastidiosa*. *Nature*, 406(6792):151–159, July 2000.

[109] STL – Standard Template Library, September 2002. www.sgi.com/tech/stl.

[110] The Sugar Cane EST Genome Project, September 2002. www.sucest.lad.ic.unicamp.br.

[111] G. P. Telles, M. D.V. Braga, Z. Dias, L. T. Li, J. A. A. Quitzau, F. R. da Silva, and J. Meidanis. Bioinformatics of the sugarcane est project. *Genetics and Molecular Biology*, 24(1-4):9–15, December 2001.

[112] G. P. Telles and F. R. da Silva. Trimming and clustering sugarcane ests. *Genetics and Molecular Biology*, 24(1-4):17–23, December 2001.

[113] The Human Cancer Genome Project, September 2002. www.ludwig.org.br/ORESTES.

[114] TIGR – The Institute for Genomic Research, September 2002. www.tigr.org.

[115] N. Q. Tran. An easy case of sorting by reversals. In A. Apostolico and J. Hein, editors, *Proceedings of the 8th Annual Symposium of the Combinatorial Pattern Matching*

(CPM'97), volume 1264 of *Lecture Notes in Computer Science*, pages 83–89, Aarhus, Denmark, June 1997. Springer.

[116] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, R. M. Ballew, D. H. Huson, J. R. Wortman, Q. Zhang, C. D. Kodira, X. H. Zheng, L. Chen, M. Skupski, G. Subramanian, P. D. Thomas, J. Zhang, G. L. G. Miklos, C. Nelson, S. Broder, A. G. Clark, J. Nadeau, V. A. McKusick, N. Zinder, A. J. Levine, R. J. Roberts, M. Simon, C. Slayman, M. Hunkapiller, R. Bolanos, A. Delcher, I. Dew, D. Fasulo, M. Flanigan, L. Florea, A. Halpern, S. Hannenhalli, S. Kravitz, S. Levy, C. Mobarry, K. Reinert, K. Remington, J. Abu-Threideh, E. Beasley, K. Biddick, V. Bonazzi, R. Brandon, M. Cargill, I. Chandramouliswaran, R. Charlab, K. Chaturvedi, Z. Deng, V. Di Francesco, P. Dunn, K. Eilbeck, C. Evangelista, A. E. Gabrielian, W. Gan, W. Ge, F. Gong, Z. Gu, P. Guan, T. J. Heiman, M. E. Higgins, R.-R. Ji, Z. Ke, K. A. Ketchum, Z. Lai, Y. Lei, Z. Li, J. Li, Y. Liang, X. Lin, F. Lu, G. V. Merkulov, N. Milshina, H. M. Moore, A. K Naik, V. A. Narayan, B. Neelam, D. Nusskern, D. B. Rusch, S. Salzberg, W. Shao, B. Shue, J. Sun, Z. Y. Wang, A. Wang, X. Wang, J. Wang, M.-H. Wei, R. Wides, C. Xiao, C. Yan, A. Yao, J. Ye, M. Zhan, W. Zhang, H. Zhang, Q. Zhao, L. Zheng, F. Zhong, W. Zhong, S. C. Zhu, S. Zhao, D. Gilbert, S. Baumhueter, G. Spier, C. Carter, A. Cravchik, T. Woodage, F. Ali, H. An, A. Awe, D. Baldwin, H. Baden, M. Barnstead, I. Barrow, K. Beeson, D. Busam, A. Carver, A. Center, M. L. Cheng, L. Curry, S. Danaher, L. Davenport, R. Desilets, S. Dietz, K. Dodson, L. Doup, S. Ferriera, N. Garg, A. Gluecksmann, B. Hart, J. Haynes, C. Haynes, C. Heiner, S. Hladun, D. Hostin, J. Houck, T. Howland, C. Ibegwam, J. Johnson, F. Kalush, L. Kline, S. Koduru, A. Love, F. Mann, D. May, S. McCawley, T. McIntosh, I. McMullen, M. Moy, L. Moy, B. Murphy, K. Nelson, C. Pfannkoch, E. Pratts, V. Puri, H. Qureshi, M. Reardon, R. Rodriguez, Y.-H. Rogers, D. Romblad, B. Ruhfel, R. Scott, C. Sitter, M. Smallwood, E. Stewart, R. Strong, E. Suh, R. Thomas, N. N. Tint, S. Tse, C. Vech, G. Wang, J. Wetter, S. Williams, M. Williams, S. Windsor, E. Winn-Deen, K. Wolfe, J. Zaveri, K. Zaveri, J. F. Abril, R. Guigó, M. J. Campbell, K. V. Sjolander, B. Karlak, A. Kejariwal, H. Mi, B. Lazareva, T. Hatton, A. Narechania, K. Diemer, A. Muruganujan, N. Guo, S. Sato, V. Bafna, S. Istrail, R. Lippert, R. Schwartz, B. Walenz, S. Yooseph, D. Allen, A. Basu, J. Baxendale, L. Blick, M. Caminha, J. Carnes-Stine, P. Caulk, Y.-H. Chiang, M. Coyne, C. Dahlke, A. D. Mays, M. Dombroski, M. Donnelly, D. Ely, S. Esparham, C. Fosler, H. Gire, S. Glanowski, K. Glasser, A. Glodek, M. Gorokhov, K. Graham, B. Gropman, M. Harris, J. Heil, S. Henderson, J. Hoover, D. Jennings, C. Jordan, J. Jordan, J. Kasha, L. Kagan, C. Kraft, A. Levitsky, M. Lewis, X. Liu, J. Lopez, D. Ma, W. Majoros, J. McDaniel, S. Murphy, M. Newman, T. Nguyen, N. Nguyen, M. Nodell, S. Pan, J. Peck, M. Peterson, W. Rowe, R. Sanders, J. Scott, M. Simpson, T. Smith, A. Sprague, T. Stockwell, R. Turner, E. Venter, M. Wang, M. Wen,

D. Wu, M. Wu, A. Xia, A. Zandieh, and X. Zhu. The sequence of the human genome. *Science*, 291(5507):1304–1351, 2001.

[117] J. P. Vergara. *Sorting by Bounded Permutations*. PhD thesis, Virginia Polytech Institute and State University, 1997.

[118] A. L. Vettore, F. R. da Silva, E. L. Kemper, G. M. Souza, A. M. da Silva, M. I. T. Ferro, F. Henrique-Silva, A. Giglioti, M. V. F. Lemos, L. L. Coutinho, M. P. Nobrega, H. Carrer, S. C. Fran, M. Bacci Jr., M. H. S. Goldman, S. L. Gomes, L. R. Nunes, L. E. A. Camargo, W. J. Siqueira, M. A. V. Sluys, O. H. Thiemann, E. E. Kuramae, R. V. Santelli, C. L. Marino, M. L. P. N. Targon, J. A. Ferro, H. C. S. Silveira, D. C. Marini, E. G. M. Lemos, C. B. Monteiro-Vitorello, J. H. M. Tambor, D. M. Carraro, P. G. Roberto, V. G. Martins, G. H. Goldman, R. C. de Oliveira, D. Truffi, C. A. Colombo, M. Rossi, P. G. de Araujo, S. A. Sculaccio, A. Angella, M. M. A. Lima, V. E. de Rosa Jr., F. Siviero, V. E. Coscrato, M. A. Machado, L. Grivet, S. M. Z. Di Mauro, F. G. Nobrega, C. F.M.Menck, M. D. V. Braga, G. P. Telles, F. A. A. Cara, G. Pedrosa, J. Meidanis, and P. Arruda. Analysis and functional annotation of an expressed sequence tag collection for the tropical crop sugarcane. To be appear in 2002.

[119] M. E. M. T. Walter. *Algoritmos para Problemas em Rearranjo de Genomas*. PhD thesis, University of Campinas, Brazil, 1999. In Portuguese.

[120] M. E. M. T. Walter, Z. Dias, and J. Meidanis. Reversal and transposition distance of linear chromosomes. In *Proceedings of the String Processing and Information Retrieval (SPIRE'98)*, 1998.

[121] M. E. M. T. Walter, Z. Dias, and J. Meidanis. A new approach for approximating the transposition distance. In *Proceedings of the String Processing and Information Retrieval (SPIRE'2000)*, September 2000.

[122] M. S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman and Hall, 1995.

[123] G. A. Watterson, W. J. Ewens, T. E. Hall, and A. Morgan. The chromosome inversion problem. *Journal of Theorical Biology*, 99:1–7, 1982.